Freeform Search

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term:

Display: 10 **Documents in Display Format:** - **Starting with Number** 1

Generate: ○ Hit List [✓ Hit Count] ○ Side by Side ○ Image

Search Clear Interrupt

Search History

DATE: Wednesday, August 01, 2007 Purge Queries Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		
L93	l57 and l84	39	L93
	DB=DWPI,TDBD; PLUR=YES; OP=OR		
L92	l57 and l84	0	L92
	DB=EPAB,JPAB; PLUR=YES; OP=OR		
L91	l57 and l84	0	L91
	DB=PGPB,USPT,USOC; PLUR=YES; OP=OR		
L90	l57 and l84	39	L90
	DB=USPT; PLUR=YES; OP=OR		
L89	'6105020'.pn.	1	L89
L88	'6397204'.pn.	1	L88
L87	'6516310'.pn.	1	L87
L86	'7010516'.pn.	1	L86
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		
L85	L84 and (customer with profile with groups or customer near profile near groups or customer adj profile adj groups)	8	L85

L84	L83 and (fact with table or fact near table or fact adj table)	81	L84
L83	l38 and (dimension with table or dimension near table or dimension adj table)	86	L83
	DB=USPT; PLUR=YES; OP=OR		
L82	(5544298 | 5088052 | 5471611 | 5414838 | 5537590 | 5455945 | 5404506)![PN]	7	L82
L81	("5832496")[PN]	1	L81
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		
L80	5832496.pn.	2	L80
L79	5832498.pn.	2	L79
L78	5615109.pn.	2	L78
	DB=USPT; PLUR=YES; OP=OR		
L77	'6438537'.pn.	1	L77
L76	'6434557'.pn.	1	L76
L75	'6374263'.pn.	1	L75
L74	'6385201'.pn.	1	L74
L73	'6477525'.pn.	1	L73
L72	'6505205'.pn.	1	L72
L71	'6505205'.pn.	1	L71
L70	'6487546'.pn.	1	L70
L69	'6484179'.pn.	1	L69
L68	'6480836'.pn.	1	L68
L67	'6173310'.pn.	1	L67
L66	'6173310'.pn.	1	L66
L65	'6151601'.pn.	1	L65
L64	'6151601'.pn.	1	L64
L63	'6141655'.pn.	1	L63
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		
L62	L57 and 717.clas.	2	L62
L61	L57 and 706.clas.	2	L61
L60	L57 and 705.clas.	19	L60
L59	L57 and 707.clas.	146	L59
L58	L57 not @py>1999	0	L58
L57	L56 and (metadata or metadata with model or meta-data near model or meta-data adj model)	168	L57
L56	(star with schema or star near schema or star adj schema or "reverse star schema" or reverse with star with schema or reverse near star near schema or snowflake with schema or snowflake near schema or snowflake adj schema) and ("data warehouse" or datamart)	324	L56
	DB=USPT; PLUR=YES; OP=OR		
L55	'5995958'.pn.	1	L55
L54	'5806060'.pn.	1	L54
L53	'5386556'.pn.	1	L53
L52	'5854746'.pn.	1	L52

L51	'5873096'.pn.	1	L51
L50	'5978788'.pn.	1	L50
L49	'6032158'.pn.	1	L49
L48	'6032158'.pn.	1	L48
L47	'6668253'.pn.	1	L47
L46	'6295551'.pn.	1	L46
L45	'6606708'.pn.	1	L45
L44	'6226752'.pn.	1	L44
L43	'6715080'.pn.	1	L43
L42	'6715080'.pn.	1	L42
L41	'6782425'.pn.	1	L41
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		
L40	L39 and (populat$ and datamart or populat$ and data near2 mart or populat$ adj data adj mart)	24	L40
L39	L38 and (business and database or business and data near2 base or business adj database or business same database)	95	L39
	DB=USPT; PLUR=YES; OP=OR		
L38	("star schema" or "reverse star schema") and (datawarehouse or data with warehouse or data with mart or datamart)	109	L38
L37	("5799286")[URPN]	74	L37
L36	("5799286")[PN]	1	L36
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		
L35	5799286.pn.	2	L35
	DB=USPT; PLUR=YES; OP=OR		
L34	("6212524")[URPN]	38	L34
L33	("6212524")[URPN]	38	L33
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		
L32	6212524.pn.	2	L32
	DB=USPT; PLUR=YES; OP=OR		
L31	("6377993")[URPN]	76	L31
L30	(5041972 | 5481542 | 5999973 | 5958016 | 5245533 | 5790797 | 6065002 | 5844896 | 5920542 | 5721913 | 6078891 | 5790780 | 5530744 | 5825769 | 5991733 | 5819225 | 5787412 | 5835084 | 5548726 | 6011844 | 5327486 | 5877759 | 5781550 | 5727129 | 5610915 | 5884032 | 5602918 | 5742905 | 5819271 | 6058381 | 6049789 | 6023762 | 4893248 | 5796393 | 5764756 | 5907681 | 5812533 | 5742768 | 5963925 | 5826269 | 4817050 | 5930764 | 5696906 | 5778377 | 5870558 | 5315093 | 5526257 | 5826029 | 5325290 | 5490060 | 5815080 | 5793964 | 5721908 | 5778178 | 6041325 | 6032132 | 5991806 | 5742763 | 5369571 | 6119109 | 5692181 | 6014647 | 5563805 | 5768501 | 6115693 | 5923016 | 6115737 | 6073105 | 5475836 | 5452446 | 5689645 | 6145001 | 5805803 | 5862325 | 5621727 | 6161128 | 6078924 | 6065059 | 6137869 | 5848396 | 5815665 | 6134584 | 5649182 | 5742762 | 4345315 | 6212506 | 5491779 | 5537611 | 5228076 | 5966695 | 6049602 | 5699403 | 5852810 | 6091808 | 6014702 | 5961602 | 6085171 | 4160129 | 5781632 | 5892900 | 5845067 | 5909682 | 5799154 | 5974396 | 5974441 | 5566351 | 5671354 | 6115458 | 6094655 | 5745754 | 5734831 | 5812654 |	156	L30

5999972 | 6253239 | 5802320 | 5131020 | 5793762 | 6044144 | 5790809 | 4972504 | 5852812 | 5790789 | 5734709 | 6031904 | 5915001 | 5623601 | 5287270 | 5754830 | 5812750 | 5551025 | 6085190 | 6212558 | 6032184 | 5774660 | 5909679 | 6131095 | 5666481 | 5692030 | 6064667 | 5930804 | 5970467 | 6128624 | 6073241 | 5706502 | 5982864 | 5483596 | 6115040 | 5075771 | 5630066 | 5699528 | 5787160 | 5708780 | 5850517 | 5982891 | 5285494 | 5845267)![PN]

L29	("6377993")[PN]	1	L29
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		
L28	6377993.pn.	2	L28
L27	717/105	553	L27
L26	717/104	891	L26
L25	717/102	271	L25
L24	706/52	645	L24
L23	705/44	1408	L23
L22	705/39	2336	L22
L21	705/35	3173	L21
L20	705/30	1335	L20
L19	705/28	2368	L19
L18	705/26	7782	L18
L17	705/16	1225	L17
L16	705/14	5636	L16
L15	705/7	3185	L15
L14	705/5	1193	L14
L13	705/1	7432	L13
L12	707/206	1595	L12
L11	707/201	4030	L11
L10	707/200	6242	L10
L9	707/104.1	8830	L9
L8	707/101	6471	L8
L7	707/100	10569	L7
L6	707/10	15344	L6
L5	707/1	9888	L5
L4	717.clas.	14525	L4
L3	706.clas.	8287	L3
L2	705.clas.	52424	L2
L1	707.clas.	57191	L1

END OF SEARCH HISTORY

IEEE Xplore® RELEASE 2.3

Search Result - Print Format

< Back t

Key: IEEE JNL = IEEE Journal or Magazine, IEE JNL = IEE Journal or Magazine, IEEE CNF = IEEE Conference, II CNF = IEE Conference, IEEE STD = IEEE Standard

1. **Dynamic multi-dimensional models for text warehouses**
   Bleyberg, M.Z.; Ganesh, K.;
   Systems, Man, and Cybernetics, 2000 IEEE International Conference on
   Volume 3, 8-11 Oct. 2000 Page(s):2045 - 2050 vol.3
   IEEE CNF

2. **Data warehouse design for manufacturing execution systems**
   Kai-Ying Chen; Teh-Chang Wu;
   Mechatronics, 2005. ICM '05. IEEE International Conference on
   10-12 July 2005 Page(s):751 - 756
   IEEE CNF

3. **TSMC turnkey data mart**
   Sung-Ting Hsieh, D.; Cheng-Chin Feng, E.; Wei-Ling Liu; I-Chieh Chung;
   Semiconductor Manufacturing Technology Workshop, 2002
   10-11 Dec. 2002 Page(s):267 - 270
   IEEE CNF

4. **A design and practical use of spatial data warehouse**
   Ji-man Park; Chul-sue Hwang;
   Geoscience and Remote Sensing Symposium, 2005. IGARSS '05. Proceedings. 2005 IEEE International
   Volume 2, 25-29 July 2005 Page(s):4 pp.
   IEEE CNF

5. **Interactive ROLAP on large datasets: a case study with UB-trees**
   Ramsak, F.; Markl, V.; Fenk, R.; Bayer, R.; Ruf, T.;
   Database Engineering & Applications, 2001 International Symposium on.
   16-18 July 2001 Page(s):167 - 176
   IEEE CNF

Indexed by Inspec®

PORTAL
USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: [✓ The ACM Digital Library] The Guide

+reverse +star +schema SEARCH

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used: reverse star schema

Found **174** of **207,474**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 174

Result page: **1** 2 3 4 5 6 7 8 9 next

Relevance scale

1 Designing data marts for data warehouses
October 2001 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 10 Issue 4
Publisher: ACM Press
Full text available: pdf(203.43 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Data warehouses are databases devoted to analytical processing. They are used to support decision-making activities in most modern business settings, when complex data sets have to be studied and analyzed. The technology for analytical processing assumes that data are presented in the form of simple data marts, consisting of a well-identified collection of facts and data analysis dimensions (star schema). Despite the wide diffusion of data warehouse technology and concepts, we still miss me ...

Keywords: conceptual modeling, data mart, data warehouse, design method, software quality management

2 Snakes and sandwiches: optimal clustering strategies for a data warehouse
H. V. Jagadish, Laks V. S. Lakshmanan, Divesh Srivastava
June 1999 **ACM SIGMOD Record , Proceedings of the 1999 ACM SIGMOD international conference on Management of data SIGMOD '99**, Volume 28 Issue 2
Publisher: ACM Press
Full text available: pdf(1.47 MB) Additional Information: full citation, abstract, references, citings, index terms

Physical layout of data is a crucial determinant of performance in a data warehouse. The optimal clustering of data on disk, for minimizing expected I/O, depends on the query workload. In practice, we often have a reasonable sense of the likelihood of different classes of queries, e.g., 40% of the queries concern calls made from some specific telephone number in some month. In this paper, we address the problem of finding an optimal clustering of records of ...

3 A comparison of data warehousing methodologies
Arun Sen, Atish P. Sinha
March 2005 **Communications of the ACM**, Volume 48 Issue 3
Publisher: ACM Press
Full text available: pdf(117.81 KB) Additional Information: full citation, abstract, references, index terms

html(28.41 KB)

Using a common set of attributes to determine which methodology to use in a particular data warehousing project.

4 Star graphics: An object-oriented implementation
Daniel E. Lipkie, Steven R. Evans, John K. Newlin, Robert L. Weissman
July 1982 **ACM SIGGRAPH Computer Graphics , Proceedings of the 9th annual conference on Computer graphics and interactive techniques SIGGRAPH '82**, Volume 16 Issue 3
Publisher: ACM Press
Full text available: pdf(955.07 KB) Additional Information: full citation, abstract, references, citings, index terms

The XEROX Star 8010 Information System features an integrated text and graphics editor. The Star hardware consists of a processor, a large bit-mapped display, a keyboard and a pointing device. Star's basic graphic elements are points, lines, rectangles, triangles, graphics frames, text frames and bar charts. The internal representation is in terms of idealized objects that are displayed or printed at resolutions determined by the output device. This paper describes the design and implementa ...

Keywords: Business graphics, Subclassing

5 Heuristic optimization of OLAP queries in multidimensionally hierarchically clustered databases
Dimitri Theodoratos, Aris Tsois
November 2001 **Proceedings of the 4th ACM international workshop on Data warehousing and OLAP DOLAP '01**
Publisher: ACM Press
Full text available: pdf(1.44 MB) Additional Information: full citation, abstract, citings, index terms

On-line analytical processing (OLAP) is a technology that encompasses applications requiring a multidimensional and hierarchical view of data. OLAP applications often require fast response time to complex grouping/aggregation queries on enormous quantities of data. Commercial relational database management systems use mainly multiple one-dimensional indexes to process OLAP queries that restrict multiple dimensions. However, in many cases, multidimensional access methods outperform one-dimensiona ...

6 Bottom-up computation of sparse and Iceberg CUBE
Kevin Beyer, Raghu Ramakrishnan
June 1999 **ACM SIGMOD Record , Proceedings of the 1999 ACM SIGMOD international conference on Management of data SIGMOD '99**, Volume 28 Issue 2
Publisher: ACM Press
Full text available: pdf(1.49 MB) Additional Information: full citation, abstract, references, citings, index terms

We introduce the Iceberg-CUBE problem as a reformulation of the datacube (CUBE) problem. The Iceberg-CUBE problem is to compute only those group-by partitions with an aggregate value (e.g., count) above some minimum support threshold. The result of Iceberg-CUBE can be used (1) to answer group-by queries with a clause such as HAVING COUNT(*) >= X, where X is greater than the threshold, (2) for mining multidimensional association rules, and (3) to complement existing strategies for identif ...

7 Automated data warehousing for rule-based CRM systems
Han-joon Kim, TaeHee Lee, Sang-goo Lee, Jonghun Chun

January 2003 **Proceedings of the 14th Australasian database conference - Volume 17 ADC '03**
Publisher: Australian Computer Society, Inc.
Full text available: pdf(274.28 KB) Additional Information: full citation, abstract, references, index terms

This paper proposes a novel way of automatically developing data warehouse configuration in rule-based CRM systems. Rule-based CRM systems assume that marketing activities are represented as a set of **IF-WHEN** rules. Currently, to provide good quality CRM functionalities, CRM systems seek to combine conventional CRM methodologies with data warehousing technology. A data warehouse can be abstractly seen as a set of materialized views. Selecting views for materialization in a data warehouse i ...

Keywords: CRM, analysis query, data warehouse, materialized view, rules, star-join index

8 Component-driven engineering of database applications
Klaus-Dieter Schewe, Bernhard Thalheim
January 2006 **Proceedings of the 3rd Asia-Pacific conference on Conceptual modelling - Volume 53 APCCM '06**
Publisher: Australian Computer Society, Inc.
Full text available: pdf(188.64 KB) Additional Information: full citation, abstract, references, index terms

Though it is commonly agreed that the design of large database schemata requires group effort, database design from component subschemata has not been investigated thoroughly. In this paper we investigate snowflake-like subschemata of database schemata expressed in the Higher-order Entity-Relationship Model (HERM). These subschemata are almost hierarchical in the sense that they may contain cycles in the schema, but not in the instances. We show that each HERM schema can be decomposed into such ...

9 Poster papers - short papers: A visual interface technique for exploring OLAP data with coordinated dimension hierarchies
Mark Sifer
November 2003 **Proceedings of the twelfth international conference on Information and knowledge management CIKM '03**
Publisher: ACM Press
Full text available: pdf(272.82 KB) Additional Information: full citation, abstract, references, index terms

Multi-dimensional data occurs in many domains while a wide variety of text based and visual interfaces for querying such data exists. But many of these interfaces are not applicable to OLAP, as they do not support use of dimension hierarchies for selection and aggregation. We introduce an interface technique which supports visual querying of OLAP data, that has been implemented in the SGViewer tool. It is based on a data graph rather than a data cube representation of the data. Our interface pre ...

Keywords: OLAP, data exploration, hierarchies, interface

10 Graphical interaction with heterogeneous databases
T. Catarci, G. Santucci, J. Cardiff
May 1997 **The VLDB Journal — The International Journal on Very Large Data Bases,** Volume 6 Issue 2
Publisher: Springer-Verlag New York, Inc.
Full text available: pdf(602.82 KB) Additional Information: full citation, abstract, citings, index terms

During the past few years our research efforts have been inspired by two different needs.

On one hand, the number of non-expert users accessing databases is growing apace. On the other, information systems will no longer be characterized by a single centralized architecture, but rather by several heterogeneous component systems. In order to address such needs we have designed a new query system with both user-oriented and multidatabase features. The system's main components are an adaptive visua ...

11 Charles W. Bachman interview: September 25-26, 2004; Tucson, Arizona
Thomas Haigh
January 2006 **ACM Oral History interviews**
Publisher: ACM Press
Full text available: pdf(761.66 KB) Additional Information: full citation, abstract

Charles W. Bachman reviews his career. Born during 1924 in Kansas, Bachman attended high school in East Lansing, Michigan before joining the Army Anti Aircraft Artillery Corp, with which he spent two years in the Southwest Pacific Theater, during World War II. After his discharge from the military, Bachman earned a B.Sc. in Mechanical Engineering in 1948, followed immediately by an M.Sc. in the same discipline, from the University of Pennsylvania. On graduation, he went to work for Do ...

12 Session 7: GYO reductions, canonical connections, tree and cyclic schemas and tree projections
Nathan Goodman, Oded Shmueli, Y. C. Tay
March 1983 **Proceedings of the 2nd ACM SIGACT-SIGMOD symposium on Principles of database systems PODS '83**
Publisher: ACM Press
Full text available: pdf(1.09 MB) Additional Information: full citation, abstract, references, citings

Database schemas may be partitioned into two sub-classes tree schemas and cyclic schemas. The analysis of tree vs cyclic schemas introduced the concepts of GYO reductions, canonical connections and tree projections. This paper investigates the intricate relationships among these concepts in the context of universal relation databases.

13 The theory of parsing, translation, and compiling
Alfred V. Aho, Jeffrey D. Ullman
January 1972 Book
Publisher: Prentice-Hall, Inc.
Full text available: pdf(98.28 MB) Additional Information: full citation, abstract, references, cited by, index terms

From volume 1 Preface (See Front Matter for full Preface)

This book is intended for a one or two semester course in compiling theory at the senior or graduate level. It is a theoretically oriented treatment of a practical subject. Our motivation for making it so is threefold.

(1) In an area as rapidly changing as Computer Science, sound pedagogy demands that courses emphasize ideas, rather than implementation details. It is our hope that the algorithms and concepts presen ...

14 Special topic section on peer to peer data management: Design issues and challenges for RDF- and schema-based peer-to-peer systems
Wolfgang Nejdl, Wolf Siberski, Michael Sintek
September 2003 **ACM SIGMOD Record**, Volume 32 Issue 3
Publisher: ACM Press

Full text available: pdf(135.94 KB) Additional Information: full citation, abstract, references, citings

Databases have employed a schema-based approach to store and retrieve structured data for decades. For peer-to-peer (P2P) networks, similar approaches are just beginning to emerge. While quite a few database techniques can be re-used in this new context, a P2P data management infrastructure poses additional challenges which have to be solved before schema-based P2P networks become as common as schema-based databases. We will describe some of these challenges and discuss approaches to solve them. ...

15 Semantics and implementation of schema evolution in object-oriented databases
Jay Banerjee, Won Kim, Hyoung-Joo Kim, Henry F. Korth
December 1987 **ACM SIGMOD Record , Proceedings of the 1987 ACM SIGMOD international conference on Management of data SIGMOD '87**, Volume 16 Issue 3
Publisher: ACM Press
Full text available: pdf(1.54 MB) Additional Information: full citation, abstract, references, citings, index terms

Object-oriented programming is well-suited to such data-intensive application domains as CAD/CAM, AI, and OIS (office information systems) with multimedia documents. At MCC we have built a prototype object-oriented database system, called ORION. It adds persistence and sharability to objects created and manipulated in applications implemented in an object-oriented programming environment. One of the important requirements of these applications is schema evolution, that is, the ability to dy ...

16 Data processing in the large: BIwTL: a business information warehouse toolkit and language for warehousing simplification and automation
Bin He, Rui Wang, Ying Chen, Ana Lelescu, James Rhodes
June 2007 **Proceedings of the 2007 ACM SIGMOD international conference on Management of data SIGMOD '07**
Publisher: ACM Press
Full text available: pdf(355.95 KB) Additional Information: full citation, abstract, references, index terms

Rapidly leveraging information analytics technologies to mine the mounting information in structured and unstructured forms, derive business insights and improve decision making is becoming increasingly critical to today's business successes. One of the key enablers of the analytics technologies is an Information Warehouse Management System (IWMS) that processes different types and forms of information, builds, and maintains the information warehouse (IW) effectively. Although traditional mul ...

Keywords: data mining, information warehouse, warehousing language

17 A graphical definition of authorization schema in the DTAC model
Jonathon E. Tidswell, John M. Potter
May 2001 **Proceedings of the sixth ACM symposium on Access control models and technologies SACMAT '01**
Publisher: ACM Press
Full text available: pdf(186.83 KB) Additional Information: full citation, abstract, references, index terms

The specification of constraint languages for access control models has proven to be difficult but remains necessary for safety and for mandatory access control policies: While the authorisation relation $(Subject \times Object \rightarrow \pow Right)$ defines the authorised permissions an authorisation schema defines how the various concepts (such as subjects, users, roles, labels) are combined to form a complete access control model.Using examples drawn from common access contr ...

Keywords: DTAC, access control, computer security, constraints, dynamic, graphs, roles, schema, type

18 Computation: finite and infinite machines
Marvin L. Minsky
January 1967 Book
Publisher: Prentice-Hall, Inc.
Additional Information: full citation, abstract, references, cited by, index terms

From the Preface (See Front Matter for full Preface)

Man has within a single generation found himself sharing the world with a strange new species: the computers and computer-like machines. Neither history, nor philosophy, nor common sense will tell us how these machines will affect us, for they do not do "work" as did machines of the Industrial Revolution. Instead of dealing with materials or energy, we are told that they handle "control" and "information" and even "intellectua ...

19 HydroJ: object-oriented pattern matching for evolvable distributed systems
Keunwoo Lee, Anthony LaMarca, Craig Chambers
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications OOPSLA '03**, Volume 38 Issue 11
Publisher: ACM Press
Full text available: pdf(311.06 KB) Additional Information: full citation, abstract, references, citings, index terms

In an evolving software system, components must be able to change independently while remaining compatible with their peers. One obstacle to independent evolution is the *brittle parameter problem*: the ability of two components to communicate can depend on a number of *inessential* details of the types, structure, and/or contents of the values communicated. If these details change, then the components can no longer communicate, even if the *essential* parts of the message remain ...

Keywords: HydroJ, XML, distributed systems, dynamic dispatch, object-oriented programming, pattern matching, semi-structured data, software evolution, ubiquitous computing

20 XML schemas: integration and translation: NeT & CoT: translating relational schemas to XML schemas using semantic constraints
Dongwon Lee, Murali Mani, Frank Chiu, Wesley W. Chu
November 2002 **Proceedings of the eleventh international conference on Information and knowledge management CIKM '02**
Publisher: ACM Press
Full text available: pdf(321.89 KB) Additional Information: full citation, abstract, references, citings, index terms

Two algorithms, called NeT and CoT, to translate relational schemas to XML schemas using various semantic constraints are presented. The XML schema representation we use is a language-independent formalism named XSchema, that is both precise and concise. A given XSchema can be mapped to a schema in any of the existing XML schema language proposals. Our proposed algorithms have the following characteristics: (1) NeT derives a nested structure from a flat relational model by repeatedly applying th ...

Keywords: XML, schema translation, semantic constraints

Results 1 - 20 of 174 Result page: **1** 2 3 4 5 6 7 8 9 next

CALIFORNIA STATE UNIVERSITY
Monterey Bay

Advanced Search :: Search Tips :: CSUMB Publisher Tips

snowflake schema CSUMB Search

Searched for **snowflake schema**. Results **1 - 1** of about **1**. Search took **0.02** seconds.

Sort by: Date / Relevance

Data Warehouse Glossary - IT@CSUMB.EDU
... **Snowflake** structure: **Snowflake** is a star **schema** with normalized dimensions. Source data: The data from the operational or legacy systems that feed the ETL ...
it.csumb.edu/departments/data/glossary.html - 59k - 2005-10-19 - Cached

snowflake schema CSUMB Search

Powered by Google

An Overview of Data Warehousing and OLAP Technology

Surajit Chaudhuri
Microsoft Research, Redmond
surajitc@microsoft.com

Umeshwar Dayal
Hewlett-Packard Labs, Palo Alto
dayal@hpl.hp.com

Abstract
Data warehousing and on-line analytical processing (OLAP) are essential elements of decision support, which has increasingly become a focus of the database industry. Many commercial products and services are now available, and all of the principal database management system vendors now have offerings in these areas. Decision support places some rather different requirements on database technology compared to traditional on-line transaction processing applications. This paper provides an overview of data warehousing and OLAP technologies, with an emphasis on their new requirements. We describe back end tools for extracting, cleaning and loading data into a data warehouse; multidimensional data models typical of OLAP; front end client tools for querying and data analysis; server extensions for efficient query processing; and tools for metadata management and for managing the warehouse. In addition to surveying the state of the art, this paper also identifies some promising research issues, some of which are related to problems that the database research community has worked on for years, but others are only just beginning to be addressed. This overview is based on a tutorial that the authors presented at the VLDB Conference, 1996.

1. Introduction
Data warehousing is a collection of *decision support* technologies, aimed at enabling the *knowledge worker* (executive, manager, analyst) to make better and faster decisions. The past three years have seen explosive growth, both in the number of products and services offered, and in the adoption of these technologies by industry. According to the *META Group*, the data warehousing market, including hardware, database software, and tools, is projected to grow from $2 billion in 1995 to $8 billion in 1998. Data warehousing technologies have been successfully deployed in many industries: manufacturing (for order shipment and customer support), retail (for user profiling and inventory management), financial services (for claims analysis, risk analysis, credit card analysis, and fraud detection), transportation (for fleet management), telecommunications (for call analysis and fraud detection), utilities (for power usage analysis), and healthcare (for outcomes analysis). This paper presents a roadmap of data warehousing technologies, focusing on the special requirements that data warehouses place on database management systems (DBMSs).

A data warehouse is a "subject-oriented, integrated, time-varying, non-volatile collection of data that is used primarily in organizational decision making."[1] Typically, the data warehouse is maintained separately from the organization's operational databases. There are many reasons for doing this. The data warehouse supports on-line analytical processing (OLAP), the functional and performance requirements of which are quite different from those of the on-line transaction processing (OLTP) applications traditionally supported by the operational databases.

OLTP applications typically automate clerical data processing tasks such as order entry and banking transactions that are the bread-and-butter day-to-day operations of an organization. These tasks are structured and repetitive, and consist of short, atomic, isolated transactions. The transactions require detailed, up-to-date data, and read or update a few (tens of) records accessed typically on their primary keys. Operational databases tend to be hundreds of megabytes to gigabytes in size. Consistency and recoverability of the database are critical, and maximizing transaction throughput is the key performance metric. Consequently, the database is designed to reflect the operational semantics of known applications, and, in particular, to minimize concurrency conflicts.

Data warehouses, in contrast, are targeted for decision support. Historical, summarized and consolidated data is more important than detailed, individual records. Since data warehouses contain consolidated data, perhaps from several operational databases, over potentially long periods of time, they tend to be orders of magnitude larger than operational databases; enterprise data warehouses are projected to be hundreds of gigabytes to terabytes in size. The workloads are query intensive with mostly ad hoc, complex queries that can access millions of records and perform a lot of scans, joins, and aggregates. Query throughput and response times are more important than transaction throughput.

To facilitate complex analyses and visualization, the data in a warehouse is typically modeled *multidimensionally*. For example, in a sales data warehouse, time of sale, sales district, salesperson, and product might be some of the dimensions of interest. Often, these dimensions are hierarchical; time of sale may be organized as a day-month-quarter-year hierarchy, product as a product-category-industry hierarchy. Typical

OLAP operations include *rollup* (increasing the level of aggregation) and *drill-down* (decreasing the level of aggregation or increasing detail) along one or more dimension hierarchies, *slice_and_dice* (selection and projection), and *pivot* (re-orienting the multidimensional view of data).

Given that operational databases are finely tuned to support known OLTP workloads, trying to execute complex OLAP queries against the operational databases would result in unacceptable performance. Furthermore, decision support requires data that might be missing from the operational databases; for instance, understanding trends or making predictions requires historical data, whereas operational databases store only current data. Decision support usually requires consolidating data from many heterogeneous sources: these might include external sources such as stock market feeds, in addition to several operational databases. The different sources might contain data of varying quality, or use inconsistent representations, codes and formats, which have to be reconciled. Finally, supporting the multidimensional data models and operations typical of OLAP requires special data organization, access methods, and implementation methods, not generally provided by commercial DBMSs targeted for OLTP. It is for all these reasons that data warehouses are implemented separately from operational databases.

Data warehouses might be implemented on standard or extended relational DBMSs, called Relational OLAP (ROLAP) servers. These servers assume that data is stored in relational databases, and they support extensions to SQL and special access and implementation methods to efficiently implement the multidimensional data model and operations. In contrast, multidimensional OLAP (MOLAP) servers are servers that directly store multidimensional data in special data structures (e.g., arrays) and implement the OLAP operations over these special data structures.

There is more to building and maintaining a data warehouse than selecting an OLAP server and defining a schema and some complex queries for the warehouse. Different architectural alternatives exist. Many organizations want to implement an integrated enterprise warehouse that collects information about all subjects (e.g., customers, products, sales, assets, personnel) spanning the whole organization. However, building an enterprise warehouse is a long and complex process, requiring extensive business modeling, and may take many years to succeed. Some organizations are settling for *data marts* instead, which are departmental subsets focused on selected subjects (e.g., a marketing data mart may include customer, product, and sales information). These data marts enable faster roll out, since they do not require enterprise-wide consensus, but they may lead to complex integration problems in the long run, if a complete business model is not developed.

In Section 2, we describe a typical data warehousing architecture, and the process of designing and operating a data warehouse. In Sections 3-7, we review relevant technologies for loading and refreshing data in a data warehouse, warehouse servers, front end tools, and warehouse management tools. In each case, we point out what is different from traditional database technology, and we mention representative products. In this paper, we do not intend to provide comprehensive descriptions of all products in every category. We encourage the interested reader to look at recent issues of trade magazines such as *Databased Advisor*, *Database Programming and Design*, *Datamation*, and *DBMS Magazine*, and vendors' Web sites for more details of commercial products, white papers, and case studies. The OLAP Council[2] is a good source of information on standardization efforts across the industry, and a paper by Codd, et al.[3] defines twelve rules for OLAP products. Finally, a good source of references on data warehousing and OLAP is the Data Warehousing Information Center[4].

Research in data warehousing is fairly recent, and has focused primarily on query processing and view maintenance issues. There still are many open research problems. We conclude in Section 8 with a brief mention of these issues.

2. Architecture and End-to-End Process

Figure 1 shows a typical data warehousing architecture.

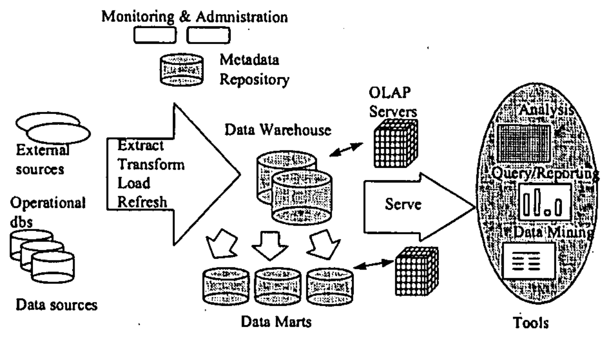


Figure 1. Data Warehousing Architecture

It includes tools for extracting data from multiple operational databases and external sources; for cleaning, transforming and integrating this data; for loading data into the data warehouse; and for periodically refreshing the warehouse to reflect updates at the sources and to purge data from the warehouse, perhaps onto slower archival storage. In addition to the main warehouse, there may be several departmental data marts. Data in the warehouse and data marts is stored and managed by one or more warehouse servers, which present multidimensional views of data to a variety of front end tools: query tools, report writers, analysis tools, and data mining tools. Finally, there is a repository for storing and managing metadata, and tools for monitoring and administering the warehousing system.

The warehouse may be distributed for load balancing, scalability, and higher availability. In such a distributed architecture, the metadata repository is usually replicated with each fragment of the warehouse, and the entire warehouse is administered centrally. An alternative architecture, implemented for expediency when it may be too expensive to construct a single logically integrated enterprise warehouse, is a federation of warehouses or data marts, each with its own repository and decentralized administration.

Designing and rolling out a data warehouse is a complex process, consisting of the following activities[5].

- Define the architecture, do capacity planning, and select the storage servers, database and OLAP servers, and tools.
- Integrate the servers, storage, and client tools.
- Design the warehouse schema and views.
- Define the physical warehouse organization, data placement, partitioning, and access methods.
- Connect the sources using gateways, ODBC drivers, or other wrappers.
- Design and implement scripts for data extraction, cleaning, transformation, load, and refresh.
- Populate the repository with the schema and view definitions, scripts, and other metadata.
- Design and implement end-user applications.
- Roll out the warehouse and applications.

3. Back End Tools and Utilities

Data warehousing systems use a variety of data extraction and cleaning tools, and load and refresh utilities for populating warehouses. Data extraction from "foreign" sources is usually implemented via gateways and standard interfaces (such as Information Builders EDA/SQL, ODBC, Oracle Open Connect, Sybase Enterprise Connect, Informix Enterprise Gateway).

Data Cleaning

Since a data warehouse is used for decision making, it is important that the data in the warehouse be correct. However, since large volumes of data from multiple sources are involved, there is a high probability of errors and anomalies in the data.. Therefore, tools that help to detect data anomalies and correct them can have a high payoff. Some examples where data cleaning becomes necessary are: inconsistent field lengths, inconsistent descriptions, inconsistent value assignments, missing entries and violation of integrity constraints. Not surprisingly, optional fields in data entry forms are significant sources of inconsistent data.

There are three related, but somewhat different, classes of data cleaning tools. *Data migration* tools allow simple transformation rules to be specified; e.g., "replace the string *gender* by *sex*". Warehouse Manager from Prism is an example of a popular tool of this kind. *Data scrubbing* tools use domain-specific knowledge (e.g., postal addresses) to do the scrubbing of data. They often exploit parsing and fuzzy matching techniques to accomplish cleaning from multiple sources. Some tools make it possible to specify the "relative cleanliness" of sources. Tools such as Integrity and Trillum fall in this category. *Data auditing* tools make it possible to discover rules and relationships (or to signal violation of stated rules) by scanning data. Thus, such tools may be considered variants of data mining tools. For example, such a tool may discover a suspicious pattern (based on statistical analysis) that a certain car dealer has never received any complaints.

Load

After extracting, cleaning and transforming, data must be loaded into the warehouse. Additional preprocessing may still be required: checking integrity constraints; sorting; summarization, aggregation and other computation to build the derived tables stored in the warehouse; building indices and other access paths; and partitioning to multiple target storage areas. Typically, batch load utilities are used for this purpose. In addition to populating the warehouse, a load utility must allow the system administrator to monitor status, to cancel, suspend and resume a load, and to restart after failure with no loss of data integrity.

The load utilities for data warehouses have to deal with much larger data volumes than for operational databases. There is only a small time window (usually at night) when the warehouse can be taken offline to refresh it. Sequential loads can take a very long time, e.g., loading a terabyte of data can take weeks and months! Hence, pipelined and partitioned parallelism are typically exploited [6]. Doing a full load has the advantage that it can be treated as a long batch transaction that builds up a new database. While it is in progress, the current database can still support queries; when the load transaction commits, the current database is replaced with the new one. Using periodic checkpoints ensures that if a failure occurs during the load, the process can restart from the last checkpoint.

However, even using parallelism, a full load may still take too long. Most commercial utilities (e.g., RedBrick Table Management Utility) use incremental loading during refresh to reduce the volume of data that has to be incorporated into the warehouse. Only the updated tuples are inserted. However, the load process now is harder to manage. The incremental load conflicts with ongoing queries, so it is treated as a sequence of shorter transactions (which commit periodically, e.g., after every 1000 records or every few seconds), but now this sequence of transactions has to be coordinated to ensure consistency of derived data and indices with the base data.

Refresh
Refreshing a warehouse consists in propagating updates on source data to correspondingly update the base data and derived data stored in the warehouse. There are two sets of issues to consider: *when* to refresh, and *how* to refresh. Usually, the warehouse is refreshed periodically (e.g., daily or weekly). Only if some OLAP queries need current data (e.g., up to the minute stock quotes), is it necessary to propagate every update. The refresh policy is set by the warehouse administrator, depending on user needs and traffic, and may be different for different sources.

Refresh techniques may also depend on the characteristics of the source and the capabilities of the database servers. Extracting an entire source file or database is usually too expensive, but may be the only choice for legacy data sources. Most contemporary database systems provide replication servers that support incremental techniques for propagating updates from a primary database to one or more replicas. Such replication servers can be used to incrementally refresh a warehouse when the sources change. There are two basic replication techniques: data shipping and transaction shipping.

In data shipping (e.g., used in the Oracle Replication Server, Praxis OmniReplicator), a table in the warehouse is treated as a remote snapshot of a table in the source database. *After_row* triggers are used to update a snapshot log table whenever the source table changes; and an automatic refresh schedule (or a manual refresh procedure) is then set up to propagate the updated data to the remote snapshot.

In transaction shipping (e.g., used in the Sybase Replication Server and Microsoft SQL Server), the regular transaction log is used, instead of triggers and a special snapshot log table. At the source site, the transaction log is sniffed to detect updates on replicated tables, and those log records are transferred to a replication server, which packages up the corresponding transactions to update the replicas. Transaction shipping has the advantage that it does not require triggers, which can increase the workload on the operational source databases. However, it cannot always be used easily across DBMSs from different vendors, because there are no standard APIs for accessing the transaction log.

Such replication servers have been used for refreshing data warehouses. However, the refresh cycles have to be properly chosen so that the volume of data does not overwhelm the incremental load utility.

In addition to propagating changes to the base data in the warehouse, the derived data also has to be updated correspondingly. The problem of constructing logically correct updates for incrementally updating derived data (materialized views) has been the subject of much research [7] [8] [9] [10]. For data warehousing, the most significant classes of derived data are summary tables, single-table indices and join indices.

4. Conceptual Model and Front End Tools

A popular conceptual model that influences the front-end tools, database design, and the query engines for OLAP is the *multidimensional* view of data in the warehouse. In a multidimensional data model, there is a set of *numeric measures* that are the objects of analysis. Examples of such measures are sales, budget, revenue, inventory, ROI (return on investment). Each of the numeric measures depends on a set of *dimensions,* which provide the context for the measure. For example, the dimensions associated with a sale amount can be the city, product name, and the date when the sale was made. The dimensions together are assumed to *uniquely* determine the measure. Thus, the multidimensional data views a measure as a value in the multidimensional space of dimensions. Each dimension is described by a set of attributes. For example, the Product dimension may consist of four attributes: the category and the industry of the product, year of its introduction, and the average profit margin. For example, the soda Surge belongs to the category beverage and the food industry, was introduced in 1996, and may have an average profit margin of 80%. The attributes of a dimension may be related via a hierarchy of relationships. In the above example, the product name is related to its category and the industry attribute through such a hierarchical relationship.

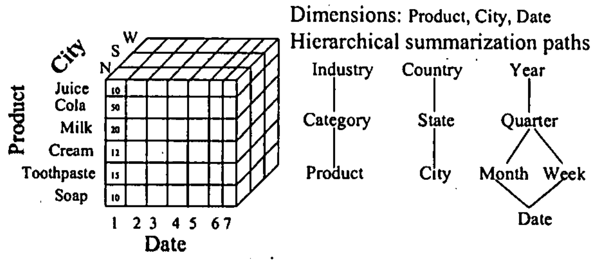


Figure 2. Multidimensional data

Another distinctive feature of the conceptual model for OLAP is its stress on *aggregation* of measures by one or more dimensions as one of the key operations; e.g., computing and ranking the *total* sales by each county (or by each year). Other popular operations include *comparing* two measures (e.g., sales and budget) aggregated by the same dimensions. Time is a dimension that is of particular significance to decision support (e.g., trend analysis). Often, it is desirable to have built-in knowledge of calendars and other aspects of the time dimension.

Front End Tools

The multidimensional data model grew out of the view of business data popularized by PC spreadsheet programs that were extensively used by business analysts. The spreadsheet is still the most compelling front-end application for OLAP. The challenge in supporting a query environment for OLAP can be crudely summarized as that of supporting spreadsheet operations efficiently over large multi-gigabyte databases. Indeed, the Essbase product of Arbor Corporation uses Microsoft Excel as the front-end tool for its multidimensional engine.

We shall briefly discuss some of the popular operations that are supported by the multidimensional spreadsheet applications. One such operation is *pivoting*. Consider the multidimensional schema of Figure 2 represented in a spreadsheet where each row corresponds to a sale . Let there be one column for each dimension and an extra column that represents the amount of sale. The simplest view of pivoting is that it selects two dimensions that are used to aggregate a measure, e.g., sales in the above example. The aggregated values are often displayed in a grid where each value in the (x,y) coordinate corresponds to the aggregated value of the measure when the first dimension has the value x and the second dimension has the value y. Thus, in our example, if the selected dimensions are city and year, then the x-axis may represent all values of city and the y-axis may represent the years. The point (x,y) will represent the aggregated sales for city x in the year y. Thus, what were values in the original spreadsheets have now become row and column headers in the pivoted spreadsheet.

Other operators related to pivoting are *rollup* or *drill-down*. Rollup corresponds to taking the current data object and doing a further group-by on one of the dimensions. Thus, it is possible to roll-up the sales data, perhaps already aggregated on city, additionally by product. The drill-down operation is the converse of rollup. *Slice_and_dice* corresponds to reducing the dimensionality of the data, i.e., taking a projection of the data on a subset of dimensions for selected values of the other dimensions. For example, we can slice_and_dice sales data for a specific product to create a table that consists of the dimensions city and the day of sale. The other popular operators include *ranking* (sorting), *selections* and defining *computed* attributes.

Although the multidimensional spreadsheet has attracted a lot of interest since it empowers the end user to analyze business data, this has not replaced traditional analysis by means of a *managed query environment*. These environments use stored procedures and predefined complex queries to provide packaged analysis tools. Such tools often make it possible for the end-user to query in terms of domain-specific business data. These applications often use raw data access tools and optimize the access patterns depending on the back end database server. In addition, there are query environments (e.g., Microsoft Access) that help build *ad hoc* SQL queries by "pointing-and-clicking". Finally, there are a variety of data mining tools that are often used as front end tools to data warehouses.

5. Database Design Methodology

The multidimensional data model described above is implemented directly by MOLAP servers. We will describe these briefly in the next section. However, when a relational ROLAP server is used, the multidimensional model and its operations have to be mapped into relations and SQL queries. In this section, we describe the design of relational database schemas that reflect the multidimensional views of data.

Entity Relationship diagrams and normalization techniques are popularly used for database design in OLTP environments. However, the database designs recommended by ER diagrams are inappropriate for decision support systems where efficiency in querying and in loading data (including incremental loads) are important.

Most data warehouses use a *star schema* to represent the multidimensional data model. The database consists of a single fact table and a single table for each dimension. Each tuple in the fact table consists of a pointer (foreign key - often uses a generated key for efficiency) to each of the dimensions that provide its multidimensional coordinates, and stores the numeric measures for those coordinates. Each dimension table consists of columns that correspond to attributes of the dimension. Figure 3 shows an example of a star schema.

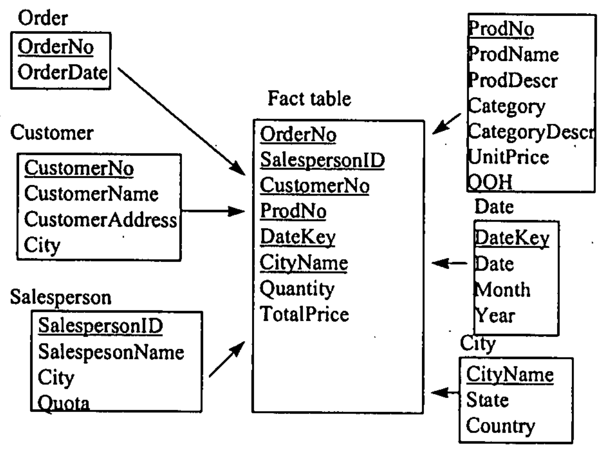


Figure 3. A Star Schema.

Star schemas do not explicitly provide support for attribute hierarchies. *Snowflake schemas* provide a refinement of star schemas where the dimensional hierarchy is explicitly represented by normalizing the dimension tables, as shown in Figure 4. This leads to advantages in maintaining the dimension tables. However, the denormalized structure of the dimensional tables in star schemas may be more appropriate for browsing the dimensions.

Fact constellations are examples of more complex structures in which multiple fact tables share dimensional tables. For example, projected expense and the actual expense may form a fact constellation since they share many dimensions.

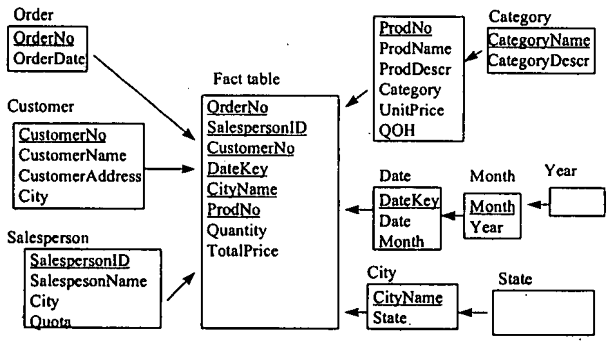


Figure 4. A Snowflake Schema.

In addition to the fact and dimension tables, data warehouses store selected summary tables containing pre-aggregated data. In the simplest cases, the pre-aggregated data corresponds to aggregating the fact table on one or more selected dimensions. Such pre-aggregated summary data can be represented in the database in at least two ways. Let us consider the example of a summary table that has total sales by product by year in the context of the star schema of Figure 3. We can represent such a summary table by *a separate fact table* which shares the dimension Product and also *a separate shrunken dimension table* for time, which consists of only the attributes of the dimension that make sense for the summary table (i.e., year). Alternatively, we can represent the summary table by encoding the aggregated tuples in the *same fact table* and the *same dimension tables* without adding new tables. This may be accomplished by adding a new *level* field to each dimension and using *nulls*: We can encode a day, a month or a year in the Date dimension table as follows: (id0, 0, 22, 01, 1960) represents a record for Jan 22, 1960, (id1, 1, NULL, 01, 1960) represents the month Jan 1960 and (id2, 2, NULL, NULL, 1960) represents the year 1960. The second attribute represents the new attribute *level*: 0 for days, 1 for months, 2 for years. In the fact table, a record containing the foreign key id2 represents the aggregated sales for a Product in the year 1960. The latter method, while reducing the number of tables, is often a source of operational errors since the level field needs be carefully interpreted.

6. Warehouse Servers

Data warehouses may contain large volumes of data. To answer queries efficiently, therefore, requires highly efficient access methods and query processing techniques. Several issues arise. First, data warehouses use redundant structures such as indices and materialized views. Choosing which indices to build and which views to materialize is an important physical design problem. The next challenge is to effectively use the existing indices and materialized views to answer queries. Optimization of complex queries is another important problem. Also, while for data-selective queries, efficient index scans may be very effective, data-intensive queries need the use of sequential scans. Thus, improving the efficiency of scans is important. Finally, parallelism needs to be exploited to reduce query response times. In this short paper, it is not possible to elaborate on each of these issues. Therefore, we will only briefly touch upon the highlights.

Index Structures and their Usage
A number of query processing techniques that exploit indices are useful. For instance, the selectivities of multiple conditions can be exploited through *index intersection.* Other useful index operations are union of indexes. These index operations can be used to significantly reduce and in many cases eliminate the need to access the base tables.

Warehouse servers can use *bit map indices*, which support efficient index operations (e.g., union, intersection). Consider a leaf page in an index structure corresponding to a domain value *d.* Such a leaf page traditionally contains a list of the record ids (RIDs) of records that contain the value *d.* However, bit map indices use an alternative representation of the above RID list as a bit vector that has one bit for each record, which is set when the domain value for that record is *d.* In a sense, the bit map index is not a new index structure, but simply an alternative representation of the RID list. The popularity of the bit map index is due to the fact that the bit vector representation of the RID lists can speed up index intersection, union, join, and aggregation[11]. For example, if we have a query of the form column1 = d & column2 = d', then we can identify the qualifying records by taking the AND of the two bit vectors. While such representations can be very useful for low cardinality domains (e.g., gender), they can also be effective for higher cardinality domains through compression of bitmaps (e.g., run length encoding). Bitmap indices were originally used in Model 204, but many products support them today (e.g., Sybase IQ). An interesting question is to decide on which attributes to index. In general, this is really a question that must be answered by the physical database design process.

In addition to indices on single tables, the specialized nature of star schemas makes *join indices* especially attractive for decision support. While traditionally indices map the value in a column to a list of rows with that value, a join index maintains the relationships between a foreign key with its matching primary keys. In the context of a star schema, a join index can relate the values of one or more attributes of a dimension table to matching rows in the fact table. For example, consider the schema of Figure 3. There can be a join index on City that maintains, for each city, a list of RIDs of the tuples in the fact table that correspond to sales in that city. Thus a join index essentially precomputes a binary join. Multikey join indices can represent precomputed n-way joins. For example, over the Sales database it is possible to construct a multidimensional join index from (Cityname, Productname) to the fact table. Thus, the index entry for (Seattle, jacket) points to RIDs of those tuples in the Sales table that have the above combination. Using such a multidimensional join index can sometimes provide savings over taking the intersection of separate indices on Cityname and Productname. Join indices can be used with bitmap representations for the RID lists for efficient join processing[12].

Finally, decision support databases contain a significant amount of descriptive text and so indices to support text search are useful as well.

Materialized Views and their Usage

Many queries over data warehouses require summary data, and, therefore, use aggregates. Hence, in addition to indices, materializing summary data can help to accelerate many common queries. For example, in an investment environment, a large majority of the queries may be based on the performance of the most recent quarter and the current fiscal year. Having summary data on these parameters can significantly speed up query processing.

The challenges in exploiting materialized views are not unlike those in using indices: (a) identify the views to materialize, (b) exploit the materialized views to answer queries, and (c) efficiently update the materialized views during load and refresh. The currently adopted industrial solutions to these problems consider materializing views that have a relatively simple structure. Such views consist of joins of the fact table with a subset of dimension tables (possibly after some selections on those dimensions), with the aggregation of one or more measures grouped by a set of attributes from the dimension tables. The structure of these views is a little more complex when the underlying schema is a snowflake.

Despite the restricted form, there is still a wide choice of views to materialize. The selection of views to materialize must take into account workload characteristics, the costs for incremental update, and upper bounds on storage requirements. Under simplifying assumptions, a greedy algorithm was shown to have good performance[13]. A related problem that underlies optimization as well as choice of materialized views is that of estimating the effect of aggregation on the cardinality of the relations.

A simple, but extremely useful, strategy for using a materialized view is to use *selection* on the materialized view, or *rollup* on the materialized view by grouping and aggregating on additional columns. For example, assume that a materialized view contains the total sales by quarter for each product. This materialized view can be used to answer a query that requests the total sales of Levi's jeans for the year by first applying the selection and then rolling up from quarters to years. It should be emphasized that the ability to do roll-up from a partially aggregated result, relies on algebraic properties of the aggregating functions (e.g., *Sum* can be rolled up, but some other statistical function may not be).

In general, there may be several candidate materialized views that can be used to answer a query. If a view V has the same set of dimensions as Q, if the selection clause in Q implies the selection clause in V, and if the group-by columns in V are a subset of the group-by columns in Q, then view V can act as a *generator* of Q. Given a set of materialized views M, a query Q, we can define a set of *minimal generators* M' for Q (i.e., smallest set of generators such that all other generators generate some member of M'). There can be multiple minimal generators for a query. For example, given a query that asks for total sales of clothing in Washington State, the following two views are both generators: (a) total sales by each state for each product (b) total sales by each city for each category. The notion of minimal generators can be used by the optimizer to narrow the search for the appropriate materialized view to use. On the commercial side, HP Intelligent Warehouse pioneered the use of the minimal generators to answer queries. While we have defined the notion of a generator in a restricted way, the general problem of optimizing queries in the presence of multiple materialized views is more complex. In the special case of Select-Project-Join queries, there has been some work in this area.[14] [15] [16]

Transformation of Complex SQL Queries

The problem of finding efficient techniques for processing complex queries has been of keen interest in query optimization. In a way, decision support systems provide a testing ground for some of the ideas that have been studied before. We will only summarize some of the key contributions.

There has been substantial work on "unnesting" complex SQL queries containing *nested subqueries* by translating them into single block SQL queries when certain syntactic restrictions are satisfied[17] [18] [19] [20]. Another direction that has been pursued in optimizing nested subqueries is reducing the number of invocations and batching invocation of inner subqueries by semi-join like techniques[21] [22]. Likewise, the problem of flattening queries containing views has been a topic of interest. The case where participating views are SPJ queries is well understood. The problem is more complex when one or more of the views contain aggregation[23]. Naturally, this problem is closely related to the problem of commuting group-by and join operators. However, commuting group-by and join is applicable in the context of single block SQL queries as well.[24] [25] [26] An overview of the field appears in a recent paper[27].

Parallel Processing

Parallelism plays a significant role in processing massive databases. Teradata pioneered some of the key technology.

All major vendors of database management systems now offer data partitioning and parallel query processing technology. The article by Dewitt and Gray provides an overview of this area[28] . One interesting technique relevant to the read-only environment of decision support systems is that of piggybacking scans requested by multiple queries (used in Redbrick). Piggybacking scan reduces the total work as well as response time by overlapping scans of multiple concurrent requests.

Server Architectures for Query Processing

Traditional relational servers were not geared towards the intelligent use of indices and other requirements for supporting multidimensional views of data. However, all relational DBMS vendors have now moved rapidly to support these additional requirements. In addition to the traditional relational servers, there are three other categories of servers that were developed specifically for decision support.

- *Specialized SQL Servers:* Redbrick is an example of this class of servers. The objective here is to provide advanced query language and query processing support for SQL queries over star and snowflake schemas in read-only environments.
- *ROLAP Servers:* These are intermediate servers that sit between a relational back end server (where the data in the warehouse is stored) and client front end tools. Microstrategy is an example of such servers. They extend traditional relational servers with specialized middleware to efficiently support multidimensional OLAP queries, and they typically optimize for specific back end relational servers. They identify the views that are to be materialized, rephrase given user queries in terms of the appropriate materialized views, and generate multi-statement SQL for the back end server. They also provide additional services such as scheduling of queries and resource assignment (e.g., to prevent runaway queries). There has also been a trend to tune the ROLAP servers for domain specific ROLAP tools. The main strength of ROLAP servers is that they exploit the scalability and the transactional features of relational systems. However, intrinsic mismatches between OLAP-style querying and SQL (e.g., lack of sequential processing, column aggregation) can cause performance bottlenecks for OLAP servers.
- *MOLAP Servers:* These servers directly support the multidimensional view of data through a multidimensional storage engine. This makes it possible to implement front-end multidimensional queries on the storage layer through direct mapping. An example of such a server is Essbase (Arbor). Such an approach has the advantage of excellent indexing properties, but provides poor storage utilization, especially when the data set is sparse. Many MOLAP servers adopt a 2-level storage representation to adapt to sparse data sets and use compression extensively. In the two-level storage representation, a set of one or two dimensional subarrays that are likely to be dense are identified, through the use of design tools or by user input, and are represented in the array format. Then, the traditional indexing structure is used to index onto these "smaller" arrays. Many of the techniques that were devised for statistical databases appear to be relevant for MOLAP servers.

SQL Extensions

Several extensions to SQL that facilitate the expression and processing of OLAP queries have been proposed or implemented in extended relational servers. Some of these extensions are described below.

- *Extended family of aggregate functions:* These include support for *rank* and *percentile* (e.g., all products in the top 10 percentile or the top 10 products by total Sale) as well as support for a variety of functions used in financial analysis (*mean, mode, median*).
- *Reporting Features:* The reports produced for business analysis often requires aggregate features evaluated on a *time window*, e.g., *moving average*. In addition, it is important to be able to provide breakpoints and running totals. Redbrick's SQL extensions provide such primitives.
- *Multiple Group-By:* Front end tools such as multidimensional spreadsheets require grouping by different sets of attributes. This can be simulated by a set of SQL statements that require scanning the same data set multiple times, but this can be inefficient. Recently, two new operators, *Rollup* and *Cube,* have been proposed to augment SQL to address this problem[29]. Thus, *Rollup* of the list of attributes (*Product, Year, City*) over a data set results in answer sets with the following applications of group by: (a) group by (Product, Year, City) (b) group by (Product, Year), and (c) group by Product. On the other hand, given a list of k columns, the *Cube* operator provides a group-by for each of the 2^k combinations of columns. Such multiple group-by operations can be executed efficiently by recognizing

commonalties among them[30]. Microsoft SQL Server supports Cube and Rollup.

- *Comparisons:* An article by Ralph Kimball and Kevin Strehlo provides an excellent overview of the deficiencies of SQL in being able to do comparisons that are common in the business world, e.g., compare the difference between the total projected sale and total actual sale by each quarter, where projected sale and actual sale are columns of a table[31]. A straightforward execution of such queries may require multiple sequential scans. The article provides some alternatives to better support comparisons. A recent research paper also addresses the question of how to do comparisons among aggregated values by extending SQL[32].

7. Metadata and Warehouse Management

Since a data warehouse reflects the business model of an enterprise, an essential element of a warehousing architecture is metadata management. Many different kinds of metadata have to be managed. *Administrative* metadata includes all of the information necessary for setting up and using a warehouse: descriptions of the source databases, back-end and front-end tools; definitions of the warehouse schema, derived data, dimensions and hierarchies, predefined queries and reports; data mart locations and contents; physical organization such as data partitions; data extraction, cleaning, and transformation rules; data refresh and purging policies; and user profiles, user authorization and access control policies. *Business* metadata includes business terms and definitions, ownership of the data, and charging policies. *Operational* metadata includes information that is collected during the operation of the warehouse: the lineage of migrated and transformed data; the currency of data in the warehouse (active, archived or purged); and monitoring information such as usage statistics, error reports, and audit trails.

Often, a metadata repository is used to store and manage all the metadata associated with the warehouse. The repository enables the sharing of metadata among tools and processes for designing, setting up, using, operating, and administering a warehouse. Commercial examples include Platinum Repository and Prism Directory Manager.

Creating and managing a warehousing system is hard. Many different classes of tools are available to facilitate different aspects of the process described in Section 2. Development tools are used to design and edit schemas, views, scripts, rules, queries, and reports. Planning and analysis tools are used for what-if scenarios such as understanding the impact of schema changes or refresh rates, and for doing capacity planning. Warehouse management tools (e.g., HP Intelligent Warehouse Advisor, IBM Data Hub, Prism Warehouse Manager) are used for monitoring a warehouse, reporting statistics and making suggestions to the administrator: usage of partitions and summary tables, query execution times, types and frequencies of drill downs or rollups, which users or groups request which data, peak and average workloads over time, exception reporting, detecting runaway queries, and other quality of service metrics. System and network management tools (e.g., HP OpenView, IBM NetView, Tivoli) are used to measure traffic between clients and servers, between warehouse servers and operational databases, and so on. Finally, only recently have workflow management tools been considered for managing the extract-scrub-transform-load-refresh process. The steps of the process can invoke appropriate scripts stored in the repository, and can be launched periodically, on demand, or when specified events occur. The workflow engine ensures successful completion of the process, persistently records the success or failure of each step, and provides failure recovery with partial roll back , retry, or roll forward.

8. Research Issues

We have described the substantial technical challenges in developing and deploying decision support systems. While many commercial products and services exist, there are still several interesting avenues for research. We will only touch on a few of these here.

Data cleaning is a problem that is reminiscent of heterogeneous data integration, a problem that has been studied for many years. But here the emphasis is on *data* inconsistencies instead of schema inconsistencies. Data cleaning, as we indicated, is also closely related to data mining, with the objective of suggesting possible inconsistencies.

The problem of physical design of data warehouses should rekindle interest in the well-known problems of index selection, data partitioning and the selection of materialized views. However, while revisiting these problems, it is important to recognize the special role played by aggregation. Decision support systems already provide the field of query optimization with increasing challenges in the traditional questions of selectivity estimation and cost-based algorithms that can exploit transformations without exploding the search space (there are plenty of transformations, but few reliable cost estimation techniques and few smart cost-based algorithms/search strategies to exploit them). Partitioning the functionality of the query engine between the middleware (e.g., ROLAP layer) and the back end server is also an interesting problem.

The management of data warehouses also presents new challenges. Detecting runaway queries, and managing and scheduling resources are problems that are important but have not been well solved. Some work has been done on the logical correctness of incrementally updating materialized views, but the performance, scalability, and recoverability properties of these techniques have not been investigated. In particular, failure and checkpointing issues in load and refresh in the presence of many indices and materialized views needs further research. The adaptation and use of workflow technology might help, but this needs further investigation.

Some of these areas are being pursued by the research community[33] [34], but others have received only cursory attention, particularly in relationship to data warehousing.

Acknowledgement

We thank Goetz Graefe for his comments on the draft.

References

[1] Inmon, W.H., *Building the Data Warehouse*. John Wiley, 1992.

[2] http://www.olapcouncil.org

[3] Codd, E.F., S.B. Codd, C.T. Salley, "Providing OLAP (On-Line Analytical Processing) to User Analyst: An IT Mandate." Available from Arbor Software's web site http://www.arborsoft.com/OLAP.html.

[4] http://pwp.starnetinc.com/larryg/articles.html

[5] Kimball, R. *The Data Warehouse Toolkit*. John Wiley, 1996.

[6] Barclay, T., R. Barnes, J. Gray, P. Sundaresan, "Loading Databases using Dataflow Parallelism." *SIGMOD Record*, Vol. 23, No. 4, Dec.1994.

[7] Blakeley, J.A., N. Coburn, P. Larson. "Updating Derived Relations: Detecting Irrelevant and Autonomously Computable Updates." *ACM TODS*, Vol.4, No. 3, 1989.

[8] Gupta, A., I.S. Mumick, "Maintenance of Materialized Views: Problems, Techniques, and Applications." *Data Eng. Bulletin*, Vol. 18, No. 2, June 1995.

[9] Zhuge, Y., H. Garcia-Molina, J. Hammer, J. Widom, "View Maintenance in a Warehousing Environment, *Proc. of SIGMOD Conf.*, 1995.

[10] Roussopoulos, N., et al., "The Maryland ADMS Project: Views R Us." *Data Eng. Bulletin*, Vol. 18, No.2, June 1995.

[11] O'Neil P., Quass D. "Improved Query Performance with Variant Indices", To appear in *Proc. of SIGMOD Conf.*, 1997.

[12] O'Neil P., Graefe G. "Multi-Table Joins through Bitmapped Join Indices" *SIGMOD Record*, Sep 1995.

[13] Harinarayan V., Rajaraman A., Ullman J.D. " Implementing Data Cubes Efficiently" *Proc. of SIGMOD Conf.*, 1996.

[14] Chaudhuri S., Krishnamurthy R., Potamianos S., Shim K. "Optimizing Queries with Materialized Views" *Intl. Conference on Data Engineering*, 1995.

[15] Levy A., Mendelzon A., Sagiv Y. "Answering Queries Using Views" *Proc. of PODS*, 1995.

[16] Yang H.Z., Larson P.A. "Query Transformations for PSJ Queries", *Proc. of VLDB*, 1987.

[17] Kim W. "On Optimizing a SQL-like Nested Query" *ACM TODS*, Sep 1982.

[18] Ganski,R., Wong H.K.T., "Optimization of Nested SQL Queries Revisited " *Proc. of SIGMOD Conf.*, 1987.

[19] Dayal, U., "Of Nests and Trees: A Unified Approach to Processing Queries that Contain Nested Subqueries, Aggregates and Quantifiers" *Proc. VLDB Conf.*, 1987.

[20] Murlaikrishna, "Improved Unnesting Algorithms for Join Aggregate SQL Queries" *Proc. VLDB Conf.*, 1992.

[21] Seshadri P., Pirahesh H., Leung T. "Complex Query Decorrelation" *Intl. Conference on Data Engineering*, 1996.

[22] Mumick I.S., Pirahesh H. "Implementation of Magic Sets in Starburst" *Proc.of SIGMOD Conf.*, 1994.

[23] Chaudhuri S., Shim K. "Optimizing Queries with Aggregate Views", *Proc. of EDBT*, 1996.

[24] Chaudhuri S., Shim K. "Including Group By in Query Optimization", *Proc. of VLDB*, 1994.

[25] Yan P., Larson P.A. "Eager Aggregation and Lazy Aggregation", *Proc. of VLDB*, 1995.

[26] Gupta A., Harinarayan V., Quass D. "Aggregate-Query Processing in Data Warehouse Environments", *Proc. of VLDB*, 1995.

[27] Chaudhuri S., Shim K. "An Overview of Cost-based Optimization of Queries with Aggregates" *IEEE Data Enginering Bulletin*, Sep 1995.

[28] Dewitt D.J., Gray J. "Parallel Database Systems: The Future of High Performance Database Systems" *CACM*, June 1992.

[29] Gray J. et.al. "Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-Tab and Sub Totals" *Data Mining and Knowledge Discovery Journal*, Vol 1, No 1, 1997.

[30] Agrawal S. et.al. "On the Computation of Multidimensional Aggregates" *Proc. of VLDB Conf.*, 1996.

[31] Kimball R., Strehlo., "Why decision support fails and how to fix it", reprinted in *SIGMOD Record*, 24(3), 1995.

[32] Chatziantoniou D., Ross K. "Querying Multiple Features in Relational Databases" *Proc. of VLDB Conf.*, 1996.

[33] Widom, J. "Research Problems in Data Warehousing." *Proc. 4th Intl. CIKM Conf.*, 1995.

[34] Wu, M-C., A.P. Buchmann. "Research Issues in Data Warehousing." Submitted for publication.

PORTAL
USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: [✓ The ACM Digital Library] The Guide SEARCH

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

An overview of data warehousing and OLAP technology

Full text Pdf (102 KB)

Source **ACM SIGMOD Record** archive
Volume 26 , Issue 1 (March 1997) table of contents
Pages: 65 - 74
Year of Publication: 1997
ISSN:0163-5808

Authors Surajit Chaudhuri Microsoft Research, Redmond
Umeshwar Dayal Hewlett-Packard Labs, Palo Alto

Publisher ACM Press New York, NY, USA

Additional Information: abstract cited by index terms collaborative colleagues peer to peer

Tools and Actions: Find similar Articles Review this Article
Save this Article to a Binder Display Formats: BibTex EndNote ACM Ref

DOI Bookmark: Use this link to bookmark this Article: http://doi.acm.org/10.1145/248603.248616
What is a DOI?

↑ ABSTRACT

Data warehousing and on-line analytical processing (OLAP) are essential elements of decision support, which has increasingly become a focus of the database industry. Many commercial products and services are now available, and all of the principal database management system vendors now have offerings in these areas. Decision support places some rather different requirements on database technology compared to traditional on-line transaction processing applications. This paper provides an overview of data warehousing and OLAP technologies, with an emphasis on their new requirements. We describe back end tools for extracting, cleaning and loading data into a data warehouse; multidimensional data models typical of OLAP; front end client tools for querying and data analysis; server extensions for efficient query processing; and tools for metadata management and for managing the warehouse. In addition to surveying the state of the art, this paper also identifies some promising research issues, some of which are related to problems that the database research community has worked on for years, but others are only just beginning to be addressed. This overview is based on a tutorial that the authors presented at the VLDB Conference, 1996.

↑ CITED BY *153*

I. Petrounias , A. Assaid, Temporal web log mining using olap techniques, Proceedings of the international conference on Computational methods in sciences and engineering, p.512-519, September 12-16, 2003, Kastoria, Greece

Yon Dohn Chung , Woo Suk Yang , Myoung Ho Kim, An efficient, robust method for processing of partial top-k/bottom-k queries using the RD-Tree in OLAP, Decision Support Systems, v.43 n.2, p.313-321, March, 2007

Biswadeep Nag , David J. DeWitt, Memory allocation strategies for complex decision support queries, Proceedings of the seventh international conference on Information and knowledge management, p.116-123, November 02-07, 1998, Bethesda, Maryland, United States

Seokjin Hong , Jinuk Bae , Taewon Lee , Sukho Lee, Histogram-by: A grouping operator for continuous domains, Data & Knowledge Engineering, v.60 n.3, p.451-467, March, 2007

Anoop Singhal , Sushil Jajodia, Data warehousing and data mining techniques for intrusion detection systems, Distributed and Parallel Databases, v.20 n.2, p.149-166, September 2006

Yeow Wei Choong , Dominique Laurent , Patrick Marcel, Computing appropriate representations for multidimensional data, Data & Knowledge Engineering, v.45 n.2, p.181-203, May 2003

Anoop Singhal, Design of a data warehouse system for network/web services, Proceedings of the thirteenth ACM international conference on Information and knowledge management, November 08-13, 2004, Washington, D.C., USA

Surajit Chaudhuri , Umeshwar Dayal, Data warehousing and OLAP for decision support, ACM SIGMOD Record, v.26 n.2, p.507-508, June 1997

Nenad Jukic , Tania Neild, Designing a first-iteration data warehouse for a financial application service provider, Annals of cases on information technology, Idea Group Publishing, Hershey, PA, 2002

Jiawei Han, Data mining tasks and methods: Rule discovery: characteristic rules, Handbook of data mining and knowledge discovery, Oxford University Press, Inc., New York, NY, 2002

Yeow Wei Choong , Dominique Laurent , Patrick Marcel, Computing appropriate representations for multidimensional data, Proceedings of the 4th ACM international workshop on Data warehousing and OLAP, p.16-23, November 09-09, 2001, Atlanta, Georgia, USA

Carlos Ordonez, Vertical and horizontal percentage aggregations, Proceedings of the 2004 ACM SIGMOD international conference on Management of data, June 13-18, 2004, Paris, France

Fernando Berzal , Ignacio Blanco , Juan-Carlos Cubero , Nicolas Marin, Component-based data mining frameworks, Communications of the ACM, v.45 n.12, December 2002

Vivekanand Gopalkrishnan , Qing Li , Kamalakar Karlapalem, Efficient query processing with structural join indexing in an object relational data warehousing environment, Data warehousing and web engineering, IRM Press, Hershey, PA, 2002

Antoaneta Ivanova , Boris Rachev, Multidimensional models: constructing data CUBE, Proceedings of the 5th international conference on Computer systems and technologies, June 17-18, 2004, Rousse, Bulgaria

Surajit Chaudhuri , Umesh Dayal, Data and knowledge in database systems: multidimensional databases and online analytical processing, Handbook of data mining and knowledge discovery, Oxford University Press, Inc., New York, NY, 2002

Mark Levene , George Loizou, Why is the snowflake schema a good data warehouse design?, Information Systems, v.28 n.3, p.225-240, May 2003

Eugene Inseok Chong , Jagannathan Srinivasan , Souripriya Das , Chuck Freiwald , Aravind Yalamanchi , Mahesh Jagannath , Anh-Tuan Tran , Ramkumar Krishnan , Richard Jiang, A mapping mechanism to support bitmap index and other auxiliary structures on tables stored as primary $B^{\pm}$-trees, ACM SIGMOD Record, v.32 n.2, June 2003

Lingli Ding , Xin Zhang , Elke A. Rundensteiner, The MRE wrapper approach: enabling incremental view maintenance of data warehouses defined on multi-relation information sources, Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP, p.30-35, November 02-06, 1999, Kansas City, Missouri, United States

Antonio Badia , Matt Chanda , Bin Cao, Adding subqueries to MySQL, what does it take to have a decision-support engine?, Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP, p.49-56, November 08-08, 2002, McLean, Virginia, USA

David W. Cheung , Bo Zhou , Ben Kao , Hongjun Lu , Tak Wah Lam , Hing Fung Ting, Requirement-based data cube schema design, Proceedings of the eighth international conference on Information and knowledge management, p.162-169, November 02-06, 1999, Kansas City, Missouri, United States

Eugene Inseok Chong , Jagannathan Srinivasan , Souripriya Das , Chuck Freiwald , Aravind Yalamanchi , Mahesh Jagannath , Anh-Tuan Tran , Ramkumar Krishnan , Richard Jiang, A mapping mechanism to support bitmap index and other auxiliary structures on tables stored as primary B$^{\pm}$-trees, Proceedings of the eleventh international conference on Information and knowledge management, November 04-09, 2002, McLean, Virginia, USA

Sriram Padmanabhan , Bishwaranjan Bhattacharjee , Tim Malkemus , Leslie Cranston , Matthew Huras, Multi-dimensional clustering: a new data layout scheme in DB2, Proceedings of the 2003 ACM SIGMOD international conference on Management of data, June 09-12, 2003, San Diego, California

Andrew N. K. Chen , Paulo B. Goes , Alok Gupta , James R. Marsden, Database design in the modern organization: identifying robust structures under changing query patterns and arrival rate conditions, Decision Support Systems, v.37 n.3, p.435-447, June 2004

Satyadeep Patnaik , Marshall Meier , Brian Henderson , Joe Hickman , Brajendra Panda, Improving the performance of lineage tracing in data warehouse, Proceedings of the 1999 ACM symposium on Applied computing, p.210-215, February 28-March 02, 1999, San Antonio, Texas, United States

Riadh Ben Messaoud , Omar Boussaid , Sabine Rabaséda, A new OLAP aggregation based on the AHC technique, Proceedings of the 7th ACM international workshop on Data warehousing and OLAP, November 12-13, 2004, Washington, DC, USA

Riadh Ben Messaoud , Omar Boussaid , Sabine Loudcher Rabaséda, Evaluation of a MCA-based approach to organize data cubes, Proceedings of the 14th ACM international conference on Information and knowledge management, October 31-November 05, 2005, Bremen, Germany

Jiawei Han , Laks V. S. Lakshmanan , Raymond T. Ng, Constraint-Based, Multidimensional Data Mining, Computer, v.32 n.8, p.46-50, August 1999

Ming-Chuan Hung , Man-Lin Huang , Don-Lin Yang , Nien-Lin Hsueh, Efficient approaches for materialized views selection in a data warehouse, Information Sciences: an International Journal, v.177 n.6, p.1333-1348, March, 2007

Nebojsa Stefanovic , Jiawei Han , Krzysztof Koperski, Object-Based Selective Materialization for Efficient Implementation of Spatial Data Cubes, IEEE Transactions on Knowledge and Data Engineering, v.12 n.6, p.938-958, November 2000

Dimitri Theodoratos, Exploiting hierarchical clustering in evaluating multidimensional aggregation queries, Proceedings of the 6th ACM international workshop on Data warehousing and OLAP, November 07-07, 2003, New Orleans, Louisiana, USA

Nikos Karayannidis , Timos Sellis, SISYPHUS: the implementation of a chunk-based storage manager for OLAP data cubes, Data & Knowledge Engineering, v.45 n.2, p.155-180, May 2003

Edward Hung , David W. Cheung , Ben Kao, Optimization in Data Cube System Design, Journal of Intelligent Information Systems, v.23 n.1, p.17-45, July 2004

John Horner , Il-Yeol Song , Peter P. Chen, An analysis of additivity in OLAP systems, Proceedings of the 7th ACM international workshop on Data warehousing and OLAP, November

12-13, 2004, Washington, DC, USA

Nam Huyn, Data analysis and mining in the life sciences, ACM SIGMOD Record, v.30 n.3, September 2001

Chang-Sup Park , Myoung Ho Kim , Yoon-Joon Lee, Usability-based caching of query results in OLAP systems, Journal of Systems and Software, v.68 n.2, p.103-119, 15 November 2003

Chee-Yong Chan , Yannis E. Ioannidis, Bitmap index design and evaluation, ACM SIGMOD Record, v.27 n.2, p.355-366, June 1998

Osmar R. Zaïane , Jiawei Han , Ze-Nian Li , Jean Hou, Mining multimedia data, Proceedings of the 1998 conference of the Centre for Advanced Studies on Collaborative research, p.24, November 30-December 03, 1998, Toronto, Ontario, Canada

Jiawei Han , Jenny Y. Chiang , Sonny Chee , Jianping Chen , Qing Chen , Shan Cheng , Wan Gong , Micheline Kamber , Krzysztof Koperski , Gang Liu , Yijun Lu , Nebojsa Stefanovic , Lara Winstone , Betty B. Xia , Osmar R. Zaiane , Shuhua Zhang , Hua Zhu, DBMiner: a system for data mining in relational databases and data warehouses, Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research, p.8, November 10-13, 1997, Toronto, Ontario, Canada

Mirek Riedewald , Divyakant Agrawal , Amr El Abbadi, Dynamic multidimensional data cubes, Multidimensional databases: problems and solutions, Idea Group Publishing, Hershey, PA, 2003

L. Palopoli , L. Pontieri , G. Terracina , D. Ursino, A novel three-level architecture for large data warehouses, Journal of Systems Architecture: the EUROMICRO Journal, v.47 n.11, p.937-958, May 2002

Yingwei Cui , Jennifer Widom , Janet L. Wiener, Tracing the lineage of view data in a warehousing environment, ACM Transactions on Database Systems (TODS), v.25 n.2, p.179-227, June 2000

Amin Y. Noaman , Ken Barker, A horizontal fragmentation algorithm for the fact relation in a distributed data warehouse, Proceedings of the eighth international conference on Information and knowledge management, p.154-161, November 02-06, 1999, Kansas City, Missouri, United States

Anindya Datta , Debra VanderMeer , Krithi Ramamritham, Parallel Star Join + DataIndexes: Efficient Query Processing in Data Warehouses and OLAP, IEEE Transactions on Knowledge and Data Engineering, v.14 n.6, p.1299-1316, November 2002

Chung-Min Chen , Munir Cochinwala , Elsa Yueh, Dealing with slow-evolving fact: a case study on inventory data warehousing, Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP, p.22-29, November 02-06, 1999, Kansas City, Missouri, United States

Roberto Esposito , Rosa Meo , Marco Botta, Answering constraint-based mining queries on itemsets using previous materialized results, Journal of Intelligent Information Systems, v.26 n.1, p.95-111, January 2006

Sam Y. Sung , Zhao Li , Peng Sun, A fast filtering scheme for large database cleansing, Proceedings of the eleventh international conference on Information and knowledge management, November 04-09, 2002, McLean, Virginia, USA

Pauline LienHua Chou , Xiuzhen Zhang, Efficiently computing the top *N* averages in iceberg cubes, Proceedings of the twenty-sixth Australasian conference on Computer science: research and practice in information technology, p.101-109, February 01, 2003, Adelaide, Australia

Nenad Jukic, Modeling strategies and alternatives for data warehousing projects, Communications of the ACM, v.49 n.4, p.83-88, April 2006

Sham Prasher , Xiaofang Zhou, Multiresolution amalgamation: dynamic spatial data cube generation, Proceedings of the fifteenth Australasian database conference, p.103-111, January 01, 2004, Dunedin, New Zealand

José María Cavero , Esperanza Marcos , Mario Piattini , Adolfo Sánchez, A methodology for datawarehouse design: conceptual modeling, Data warehousing and web engineering, IRM Press, Hershey, PA, 2002

Cecil Eng Huang Chua , Roger H. L. Chiang , Ee-Peng Lim, An intelligent middleware for linear correlation discovery, Decision Support Systems, v.32 n.4, p.313-326, March 2002

Chang-Sup Park , Myoung Ho Kim , Yoon-Joon Lee, Finding an efficient rewriting of OLAP queries using materialized views in data warehouses, Decision Support Systems, v.32 n.4, p.379-399, March 2002

Swarup Acharya , Phillip B. Gibbons , Viswanath Poosala, Congressional samples for approximate answering of group-by queries, ACM SIGMOD Record, v.29 n.2, p.487-498, June 2000

Young-Koo Lee , Kyu-Young Whang , Yang-Sae Moon , Il-Yeol Song, An aggregation algorithm using a multidimensional file in multidimensional OLAP, Information Sciences: an International Journal, v.152 n.1, p.121-138, June 2003

Venkatesh Ganti , Johannes Gehrke , Raghu Ramakrishnan, DEMON: Mining and Monitoring Evolving Data, IEEE Transactions on Knowledge and Data Engineering, v.13 n.1, p.50-63, January 2001

Avigdor Gal , Vijayalakshmi Atluri, An authorization model for temporal data, Proceedings of the 7th ACM conference on Computer and communications security, p.144-153, November 01-04, 2000, Athens, Greece

Amit Rudra , Raj Gopalan, Adaptive use of a cluster of PCs for data warehousing applications: some problems and issues, Proceedings of the 2000 ACM symposium on Applied computing, p.698-703, March 2000, Como, Italy

Sara Cohen, Containment of aggregate queries, ACM SIGMOD Record, v.34 n.1, March 2005

L. Schlesinger , A. Bauer , W. Lehner , G. Ediberidze , M. Gutzmann, Efficiently synchronizing multidimensional schema data, Proceedings of the 4th ACM international workshop on Data warehousing and OLAP, p.69-76, November 09-09, 2001, Atlanta, Georgia, USA

Sergio Greco , Luigi Pontieri , Ester Zumpano, A technique for information system integration, Proceedings of the 2001 international conference on Information systems technology and its applications, p.75-84, June 13-15, 2001, Kharkiv, Ukraine

Jiawei Han , Jian Pei , Guozhu Dong , Ke Wang, Efficient computation of Iceberg cubes with complex measures, ACM SIGMOD Record, v.30 n.2, p.1-12, June 2001

Erik Riedel , Christos Faloutsos , Gregory R. Ganger , David F. Nagle, Data mining on an OLTP system (nearly) for free, ACM SIGMOD Record, v.29 n.2, p.13-21, June 2000

J. Trujillo , M. Palomar, An object oriented approach to multidimensional database conceptual modeling (OOMD), Proceedings of the 1st ACM international workshop on Data warehousing and OLAP, p.16-21, November 02-07, 1998, Washington, D.C., United States

Dimitri Theodoratos , Timos Sellis, Incremental Design of a Data Warehouse, Journal of Intelligent Information Systems, v.15 n.1, p.7-27, July—Aug. 2000

Rami Rifaieh , Nabila Aïcha Benharkat, Query-based data warehousing tool, Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP, p.35-42, November 08-08, 2002, McLean, Virginia, USA

Torsten Priebe , Günther Pernul, Towards integrative enterprise knowledge portals, Proceedings of the twelfth international conference on Information and knowledge management, November 03-08, 2003, New Orleans, LA, USA

Bongki Moon , Ines Fernando Vega Lopez , Vijaykumar Immanuel, Efficient Algorithms for Large-Scale Temporal Aggregation, IEEE Transactions on Knowledge and Data Engineering, v.15 n.3, p.744-759, March 2003

Roland Holten, Specification of management views in information warehouse projects, Information Systems, v.28 n.7, p.709-751, 1 October 2003

Luigi Palopoli , Giorgio Terracina , Domenico Ursino, Experiences using DIKE, a system for supporting cooperative information system and data warehouse design, Information Systems, v.28 n.7, p.835-865, 1 October 2003

Hao Fan , Alexandra Poulovassilis, Using AutoMed metadata in data warehousing environments, Proceedings of the 6th ACM international workshop on Data warehousing and OLAP, November 07-07, 2003, New Orleans, Louisiana, USA

Mirek Riedewald , Divyakant Agrawal , Amr El Abbadi, Efficient integration and aggregation of historical information, Proceedings of the 2002 ACM SIGMOD international conference on Management of data, June 03-06, 2002, Madison, Wisconsin

Yannis Sismanis , Antonios Deligiannakis , Yannis Kotidis , Nick Roussopoulos, Hierarchical dwarfs for the rollup cube, Proceedings of the 6th ACM international workshop on Data warehousing and OLAP, November 07-07, 2003, New Orleans, Louisiana, USA

Yannis Kotidis , Nick Roussopoulos, A case for dynamic view management, ACM Transactions on Database Systems (TODS), v.26 n.4, p.388-423, December 2001

Chien-Le Goh , Kazuki Aisaka , Masahiko Tsukamoto , Shojiro Nishio, Database compression with data mining methods, Information organization and databases: foundations of data organization, Kluwer Academic Publishers, Norwell, MA, 2000

Rajesh R. Bordawekar , Christian A. Lang, Analytical processing of XML documents: opportunities and challenges, ACM SIGMOD Record, v.34 n.2, June 2005

Ladjel Bellatreche , Arnaud Giacometti , Patrick Marcel , Hassina Mouloudi , Dominique Laurent, A personalization framework for OLAP queries, Proceedings of the 8th ACM international workshop on Data warehousing and OLAP, November 04-05, 2005, Bremen, Germany

Doron Rotem , Kurt Stockinger , Kesheng Wu, Optimizing candidate check costs for bitmap indices, Proceedings of the 14th ACM international conference on Information and knowledge management, October 31-November 05, 2005, Bremen, Germany

Gang Gou , Maxim Kormilitsin , Rada Chirkova, Query evaluation using overlapping views: completeness and efficiency, Proceedings of the 2006 ACM SIGMOD international conference on Management of data, June 27-29, 2006, Chicago, IL, USA

Rada Chirkova , Fereidoon Sadri, Query optimization using restructured views, Proceedings of the 15th ACM international conference on Information and knowledge management, November 06-11, 2006, Arlington, Virginia, USA

Anne-Muriel Arigon , Anne Tchounikine , Maryvonne Miquel, A multiversion model for multimedia data warehouse, Proceedings of the 6th international workshop on Multimedia data mining: mining integrated media and complex data, p.7-13, August 21-21, 2005, Chicago, Illinois

Marcelo Arenas , Leopoldo Bertossi , Jan Chomicki, Consistent query answers in inconsistent databases, Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, p.68-79, May 31-June 03, 1999, Philadelphia, Pennsylvania, United States

Franck Ravat , Olivier Teste , Giles Zurfluh, Towards data warehouse design, Proceedings of the eighth international conference on Information and knowledge management, p.359-366, November 02-06, 1999, Kansas City, Missouri, United States

Rónán Páircéir , Sally McClean , Bryan Scotney, Discovery of multi-level rules and exceptions from a distributed database, Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, p.523-532, August 20-23, 2000, Boston, Massachusetts, United States

Nguyen Thanh Binh , A. Min Tjoa, Conceptual multidimensional data model based on object-oriented metacube, Proceedings of the 2001 ACM symposium on Applied computing, p.295-300, March 2001, Las Vegas, Nevada, United States

Mario Cannataro , Domenico Talia , Pradip K. Srimani, Parallel data intensive computing in scientific and commercial applications, Parallel Computing, v.28 n.5, p.673-704, May 2002

Jorge R. Bernardino , Pedro S. Furtado , Henrique C. Madeira, Approximate Query Answering Using Data Warehouse Striping, Journal of Intelligent Information Systems, v.19 n.2, p.145-167, September 2002

Timo Niemi , Lasse Hirvonen , Kalervo Järvelin, Multidimensional data model and query language for informetrics, Journal of the American Society for Information Science and Technology, v.54 n.10, p.939-951, August 2003

L. Palopoli , D. Saccà , G. Terracina , D. Ursino, A technique for deriving hyponymies and overlappings from database schemes, Data & Knowledge Engineering, v.40 n.3, p.285-314, March 2002

Joachim Hammer , Lixin Fu, CubiST^{++}: Evaluating Ad-Hoc CUBE Queries Using Statistics Trees, Distributed and Parallel Databases, v.14 n.3, p.221-254, November 2003

Reda Alhajj , Mehmet Kaya, Employing OLAP mining for multiagent reinforcement learning, Design and application of hybrid intelligent systems, IOS Press, Amsterdam, The Netherlands, 2003

Guozhu Dong , Jiawei Han , Joyce M. W. Lam , Jian Pei , Ke Wang , Wei Zou, Mining Constrained Gradients in Large Databases, IEEE Transactions on Knowledge and Data Engineering, v.16 n.8, p.922-938, August 2004

Hai Wang , Kenneth C. Sevcik, A multi-dimensional histogram for selectivity estimation and fast approximate query answering, Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research, p.328-342, October 06-09, 2003, Toronto, Ontario, Canada

Jiawei Han , Yongjian Fu, Mining Multiple-Level Association Rules in Large Databases, IEEE Transactions on Knowledge and Data Engineering, v.11 n.5, p.798-805, September 1999

Kristen LeFevre , David J. DeWitt , Raghu Ramakrishnan, Incognito: efficient full-domain K-anonymity, Proceedings of the 2005 ACM SIGMOD international conference on Management of data, June 14-16, 2005, Baltimore, Maryland

Tho Manh Nguyen , Josef Schiefer , A. Min Tjoa, Sense & response service architecture (SARESA): an approach towards a real-time business intelligence solution and its use for a fraud detection application, Proceedings of the 8th ACM international workshop on Data warehousing and OLAP, November 04-05, 2005, Bremen, Germany

David W. Cheung , H. Y. Hwang , Ada W. Fu , Jiawei Han, Efficient Rule-Based Attribute-Oriented Induction for Data Mining, Journal of Intelligent Information Systems, v.15 n.2, p.175-200, Sept./Oct. 2000

Bartosz B□bel , Johann Eder , Christian Koncilia , Tadeusz Morzy , Robert Wrembel, Creation and management of versions in multiversion data warehouse, Proceedings of the 2004 ACM symposium on Applied computing, March 14-17, 2004, Nicosia, Cyprus

Kesheng Wu , Ekow J. Otoo , Arie Shoshani, Optimizing bitmap indices with efficient compression, ACM Transactions on Database Systems (TODS), v.31 n.1, p.1-38, March 2006

Rada Chirkova , Fereidoon Sadri, Query optimization using restructured views, Proceedings of the 15th ACM international conference on Information and knowledge management, November 06-11, 2006, Arlington, Virginia, USA

Ahmed E. Hassan , Richard C. Holt, Architecture recovery of web applications, Proceedings of the 24th International Conference on Software Engineering, May 19-25, 2002, Orlando, Florida

Lixin Fu , Joachim Hammer, CubiST: a new algorithm for improving the performance of ad-hoc OLAP queries, Proceedings of the 3rd ACM international workshop on Data warehousing and OLAP, p.72-79, November 06-11, 2000, McLean, Virginia, United States

Yannis Kotidis, Aggregate view management in data warehouses, Handbook of massive data sets, Kluwer Academic Publishers, Norwell, MA, 2002

Alin Dobra , Minos Garofalakis , Johannes Gehrke , Rajeev Rastogi, Processing complex aggregate queries over data streams, Proceedings of the 2002 ACM SIGMOD international conference on Management of data, June 03-06, 2002, Madison, Wisconsin

Kurt Stockinger , Kesheng Wu , Arie Shoshani, Strategies for processing ad hoc queries on large data warehouses, Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP, p.72-79, November 08-08, 2002, McLean, Virginia, USA

Marco Costa , Henrique Madeira, Handling big dimensions in distributed data warehouses using the DWS technique, Proceedings of the 7th ACM international workshop on Data warehousing and OLAP, November 12-13, 2004, Washington, DC, USA

Surajit Chaudhuri, An overview of query optimization in relational systems, Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, p.34-43, June 01-04, 1998, Seattle, Washington, United States

Nenad Jukić , Svetlozar Nestorov, Comprehensive data warehouse exploration with qualified association-rule mining, Decision Support Systems, v.42 n.2, p.859-878, November 2006

Sergio Luján-Mora , Juan Trujillo , Il-Yeol Song, A UML profile for multidimensional modeling in data warehouses, Data & Knowledge Engineering, v.59 n.3, p.725-769, December 2006

Robert M. Bruckner , A. Min Tjoa, Capturing Delays and Valid Times in Data Warehouses—Towards Timely Consistent Analyses, Journal of Intelligent Information Systems, v.19 n.2, p.169-190, September 2002

Esther Pacitti , Pascale Minet , Eric Simon, Replica Consistency in Lazy Master Replicated Databases, Distributed and Parallel Databases, v.9 n.3, p.237-267, May 2001

Sohail Asghar , Damminda Alahakoon, A hybrid neural network based DBMS system for enhanced functionality, Design and application of hybrid intelligent systems, IOS Press, Amsterdam, The Netherlands, 2003

Xinjian Lu , Franklin Lowenthal, Arranging fact table records in a data warehouse to improve query performance, Computers and Operations Research, v.31 n.13, p.2165-2182, November 2004

Sohail Asghar , Damminda Alahakoon , Arthur Hsu, Enhancing OLAP functionality using self-organizing neural networks, Neural, Parallel & Scientific Computations, v.12 n.1, p.1-20, March 2004

Zhiyong Peng , Qing Li , Ling Feng , Xuhui Li , Junqiang Liu, Using Object Deputy Model to Prepare Data for Data Warehousing, IEEE Transactions on Knowledge and Data Engineering, v.17 n.9, p.1274-1288, September 2005

Adam Fadlalla, An experimental investigation of the impact of aggregation on the performance of data mining with logistic regression, Information and Management, v.42 n.5, p.695-707, July 2005

Anne-Muriel Arigon , Anne Tchounikine , Maryvonne Miquel, Handling multiple points of view in a multimedia data warehouse, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP), v.2 n.3, p.199-218, August 06

Zina Ben-Miled , Yang Liu , Michael Bem , Robert Jones , Robert Oppelt , Samuel Milosevich , Dave Powers , Omran Bukhres, Data access performance in a large and dynamic pharmaceutical drug candidate database, Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM), p.22-es, November 04-10, 2000, Dallas, Texas, United States

Josiane Mothe , Claude Chrisment , Bernard Dousset , Joel Alaux, DocCube: multi-dimensional visualisation and exploration of large document sets, Journal of the American Society for Information Science and Technology, v.54 n.7, p.650-659, May 2003

Designing data marts for data warehouses, ACM Transactions on Software Engineering and Methodology (TOSEM), v.10 n.4, p.452-483, Oct. 2001

Ling Feng , Tharam S. Dillon, Using Fuzzy Linguistic Representations to Provide Explanatory Semantics for Data Warehouses, IEEE Transactions on Knowledge and Data Engineering, v.15 n.1, p.86-102, January 2003

Jiawei Han , Kevin Chang, Data Mining for Web Intelligence, Computer, v.35 n.11, p.64-70, November 2002

Song Lin , Donald E. Brown, An outlier-based data association method for linking criminal incidents, Decision Support Systems, v.41 n.3, p.604-615, March 2006

Jiawei Han , Yixin Chen , Guozhu Dong , Jian Pei , Benjamin W. Wah , Jianyong Wang , Y. Dora Cai, Stream Cube: An Architecture for Multi-Dimensional Analysis of Data Streams, Distributed and Parallel Databases, v.18 n.2, p.173-197, September 2005

Stéphane Grumbach , Leonardo Tininini, On the content of materialized aggregate views, Journal of Computer and System Sciences, v.66 n.1, p.133-168, 01 February 2003

Ricardo Chalmeta , Reyes Grangel, ARDIN extension for virtual enterprise integration, Journal of Systems and Software, v.67 n.3, p.141-152, 15 September 2003

Scott Nicholson, The basis for bibliomining: frameworks for bringing together usage-based data mining and bibliometrics through data warehousing in digital library services, Information Processing and Management: an International Journal, v.42 n.3, p.785-804, May 2006

Nicolas Prat , Jacky Akoka , Isabelle Comyn-Wattiau, A UML-based data warehouse design method, Decision Support Systems, v.42 n.3, p.1449-1473, December 2006

Riccardo Torlone, Conceptual multidimensional models, Multidimensional databases: problems and solutions, Idea Group Publishing, Hershey, PA, 2003

Egemen Tanin , Ben Shneiderman , Hairuo Xie, Browsing large online data tables using generalized query previews, Information Systems, v.32 n.3, p.402-423, May, 2007

Yannis Kotidis, Extending the data warehouse for service provisioning data, Data & Knowledge Engineering, v.59 n.3, p.700-724, December 2006

Satyanarayana R Valluri , Soujanya Vadapalli , Kamalakar Karlapalem, View relevance driven materialized view selection in data warehousing environment, Australian Computer Science Communications, v.24 n.2, p.187-196, January-February 2002

Marcelo Arenas , Leopoldo Bertossi, Hypothetical Temporal Reasoning in Databases, Journal of Intelligent Information Systems, v.19 n.2, p.231-259, September 2002

Surajit Chaudhuri , Vivek Narasayya , Sunita Sarawagi, Extracting predicates from mining models for efficient query evaluation, ACM Transactions on Database Systems (TODS), v.29 n.3, p.508-544, September 2004

Dong Xin , Jiawei Han , Hong Cheng , Xiaolei Li, Answering top-k queries with multi-dimensional selections: the ranking cube approach, Proceedings of the 32nd international conference on Very large data bases, September 12-15, 2006, Seoul, Korea

Nick Bassiliades , Ioannis Vlahavas , Ahmed K. Elmagarmid , Elias N. Houstis, InterBase-KB: Integrating a Knowledge Base System with a Multidatabase System for Data Warehousing, IEEE Transactions on Knowledge and Data Engineering, v.15 n.5, p.1188-1205, September 2003

Mirek Riedewald , Divyakant Agrawal , Amr El Abbadi, Managing and analyzing massive data sets with data cubes, Handbook of massive data sets, Kluwer Academic Publishers, Norwell, MA, 2002

Michael O. Akinde , Michael H. Böhlen , Theodore Johnson , Laks V. S. Lakshmanan , Divesh Srivastava, Efficient OLAP query processing in distributed data warehouses, Information Systems, v.28 n.1-2, p.111-135, 01 March 2003

Gediminas Adomavicius , Ramesh Sankaranarayanan , Shahana Sen , Alexander Tuzhilin, Incorporating contextual information in recommender systems using a multidimensional approach, ACM Transactions on Information Systems (TOIS), v.23 n.1, p.103-145, January 2005

Renata Teixeira , Aman Shaikh , Tim Griffin , Geoffrey M. Voelker, Network sensitivity to hot-potato disruptions, ACM SIGCOMM Computer Communication Review, v.34 n.4, October 2004

Juan Trujillo , Manuel Palomar , Jaime Gómez, The GOLD definition language (GDL): an object oriented formal specification language for multidimensional databases, Proceedings of the 2000 ACM symposium on Applied computing, p.346-350, March 2000, Como, Italy

Thomas Thalhammer , Michael Schrefl, Realizing active data warehouses with off-the-shelf database technology, Software—Practice & Experience, v.32 n.12, p.1193-1222, October 2002

Luigi Palopoli , Domenico Rosaci , Giorgio Terracina , Domenico Ursino, A graph-based approach for extracting terminological properties from information sources with heterogeneous formats, Knowledge and Information Systems, v.8 n.4, p.462-497, November 2005

Juan Trujillo , Manuel Palomar , Jaime Gómez, Detecting patterns and OLAP operations in the GOLD model, Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP, p.48-53, November 02-06, 1999, Kansas City, Missouri, United States

Theodore Johnson, Data warehousing, Handbook of massive data sets, Kluwer Academic Publishers, Norwell, MA, 2002

Gediminas Adomavicius , Alexander Tuzhilin, Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions, IEEE Transactions on

Knowledge and Data Engineering, v.17 n.6, p.734-749, June 2005

Leonardo Tininini, Querying multidimensional data, Multidimensional databases: problems and solutions, Idea Group Publishing, Hershey, PA, 2003

Oliver Günther, Data management in environmental information systems, Handbook of massive data sets, Kluwer Academic Publishers, Norwell, MA, 2002

Brenton Louie , Peter Mork , Fernando Martin-Sanchez , Alon Halevy , Peter Tarczy-Hornoch, Methodological Review: Data integration and genomic medicine, Journal of Biomedical Informatics, v.40 n.1, p.5-16, February, 2007

Hanan Samet, Object-based and image-based object representations, ACM Computing Surveys (CSUR), v.36 n.2, p.159-217, June 2004

Klaus R. Dittrich , Hans Fritschi , Stella Gatziu , Andreas Geppert , Anca Vaduva, SAMOS in hindsight: experiences in building an active object-oriented DBMS, Information Systems, v.28 n.5, p.369-392, July 2003

Zhiyuan Chen , Chen Li , Jian Pei , Yufei Tao , Haixun Wang , Wei Wang , Jiong Yang , Jun Yang , Donghui Zhang, Recent progress on selected topics in database research: a report by nine young Chinese researchers working in the United States, Journal of Computer Science and Technology, v.18 n.5, p.538-552, September 2003

↑ INDEX TERMS

Primary Classification:
H. Information Systems
↪ **H.4** INFORMATION SYSTEMS APPLICATIONS
↪ **H.4.2** Types of Systems
↪ **Subjects:** Decision support (e.g., MIS)

Additional Classification:
A. General Literature

General Terms:
Design, Management, Performance, Theory

↑ Collaborative Colleagues:

Surajit Chaudhuri:

Ashraf Aboulnaga
Rakesh Agrawal
Sanjay Agrawal
Brian Babcock
Roger Barga
Jeff Bernhardt
Adam Bosworth
Nicolas Bruno
Kaushik Chakrabarti
Ashok K. Chandra
Moses Charikar
Zhiyuan Chen
Zhiyuan Chen
Usama M. Fayyad
Prasanna Ganesan
Kris Ganjam
Venkatesh Ganti
Venky Ganti
Hector Garcia-Molina
Shahram Ghandeharizadeh
Luis Gravano
Jim Gray
Ashish Kumar Gupta
Waqar Hasan
Vagelis Hristidis
Seung-won Hwang
Arnd Christian König
Rahul Kapoor
Paul Larson
Andrew Layman
Guy M. Lohman
David Lomet
David B. Lomet
David Madigan
David Maier
Arun Marathe
Amelie Marian
Rajeev Motwani
Vivek Narasayya
Vivek R. Narasayya
Jeffrey F. Naughton
Amir Netz
Frank Pellow
Hamid Pirahesh
Sunita Sarawagi
Cyrus Shahabi
Kyuseok Shim
Kyuseok Shim
Abraham Silberschatz
Utkarsh Srivastava
Michael Stonebraker
Manoj Syamala
Dilys Thomas
Moshe Y. Vardi
Theo Vassilakis
Murali Venkatrao
Michael G. Walker
Gerhard Weikum

Nilesh Dalvi
Gautam Das
Mayur Datar
Umesh Dayal
Umeshwar Dayal
Klaus R. Dittrich
Usama Fayyad
Raghav Kaushik
Phokion G. Kolaitis
Lubor Kollar
Henry F. Korth
Ravi Krishnamurthy
Spyros Potamianos
Ravishankar
Ramamurthy
Don Reichart
Andreas Reuter
Gio Wiederhold
Marianne Winslett
Yuqing Wu
Yuqing Wu
Tak W. Yan

Umeshwar Dayal:

M. Tamer Özsu
Tamer Özsu
Serge Abiteboul
Rakesh Agrawal
Philip A
Bernstein
Philip A.
Bernstein
Joachim Biskup
Angela Bonifati
Alejandro P.
Buchmann
Fabio Casati
Malu Castellanos
Stefano Ceri
Arvola Chan
Ashok K.
Chandra
Surajit
Chaudhuri
Qiming Chen
Parvathi Chundi
James Clifford
Daniel Cotting
Kemal A. Delic
Klaus Dittrich
Klaus R. Dittrich
Laurent Douillet
Weimin Du
Ramez Elmasri
Stephen Fox
Venkatesh Ganti
Hector Garcia-Molina
Pankaj Garg
Nathan Goodman
Mohamed G. Gouda
Jim Gray
Daniela Grigori
Martin Griss
Martin L. Griss
Markus Gross
Jiawei Han
Eric Hanson
Ming C. Hao
Sandra Heiler
Thomas Holenstein
M. Hsu
Mei-Chun Hsu
Meichun Hsu
Hai Yann Hwang
Hai-Yann Hwang
Matthew Jacobsen
Magdi N. Kamel
Randy H. Katz
Daniel A. Keim
Johannes Klein
Ravindran
Krishnamurthy
Adrian Krug
Rivka Ladin
Terry Landers
Vijay Machiraju
Frank Manola
Dennis McCarthy
Dennis R. McCarthy
Behzad Mortazavi-
Asl
Inderpal S. Mumick
Erich J. Neuhold
M. Tamer Ozsu
Jian Pei
Barbara Pernici
Helen Pinto
Niki Pissinou
Andreas Reuter
Daniel Ries
Daniel R. Ries
Arnon Rosenthal
Mehmet Sayal
Jörn Schneidewind
Joern Schneidewind
Arie Segev
Eric Shan
Ming-Chien Shan
Dale Skeen
John M Smith
John Miles Smith
Richard Thomas
Snodgrass
Michael
Stonebraker
Babis Theodoulidis
Douglas Tolbert
Shalom Tsur
Patrick Valduriez
Moshe Y. Vardi
Gilbert Vidal
Jianyong Wang
Ke Wang
Gerhard Weikum
Jennifer Widom
Gio Wiederhold
Peter Wright
Gene T. J. Wuu
Tak W Yan
Tak W. Yan
Tak Woon Yan
Bin Zhang

Peer to Peer - Readers of this Article have also read:

- Constructing reality **Proceedings of the 11th annual international conference on Systems documentation**
  Douglas A. Powell , Norman R. Ball , Mansel W. Griffiths
- Data structures for quadtree approximation and compression **Communications of the ACM** 28, 9
  Hanan Samet
- A hierarchical single-key-lock access control using the Chinese remainder theorem **Proceedings of the 1992 ACM/SIGAPP Symposium on Applied computing**
  Kim S. Lee , Huizhu Lu , D. D. Fisher
- The GemStone object database management system **Communications of the ACM** 34, 10
  Paul Butterworth , Allen Otis , Jacob Stein
- An intelligent component database for behavioral synthesis **Proceedings of the 27th**

ACM/IEEE conference on Design automation
Gwo-Dong Chen , Daniel D. Gajski

Data Warehouse Glossary

This glossary is a compilation of definitions contributed by the experts.

Access: The act of retrieving data from the data warehouse databases.

Access Path: The path selected by the database management system to locate and retrieve requested data.

Ad hoc query: A request for information that is normally fabricated and run a single time and cannot be anticipated in advance. It consists of an SQL statement that has been constructed by a knowledgeable user or through a data access tool.

Aggregation: The process by which data values are collected with the intent to manage the collection as a single unit.
Example: The combination of fields for the same customer extracted from multiple sources.

Analysis: The act of evaluating the data retrieved from the data warehouse.

Analytics applications: Processes that produce information for management decisions, usually involving demographic analysis, trend analysis, pattern recognition, drill-down analysis and profiling.

Examples of analytics applications include: customer segmentation, customer probability models, campaign measurement, up-sell opportunities, cross-channel analysis, sales distribution analysis, cross-sell opportunities, trigger inventory analysis, supply chain analysis, customer quality analysis, channel satisfaction measurement, click stream analysis, backlog analysis, churn analysis, interaction analysis, booking analysis, billing analysis, distribution analysis, retention analysis, delivery analysis, fulfillment analysis, and promotion effectiveness.

Anomaly: A deviation, irregularity, or an unexpected result. A data anomaly may occur when a data field defined for one purpose is used for another. Examples of anomalies are negative numeric fields that should be positive (negative number of dependents), abnormally high numeric values (person weighing 3000 pounds), pairs of values in related columns that make no sense (male patient having a hysterectomy).

Architect: A person or team who defines how the environment for the data warehouse, analytics application, or operational system is built.

Architecture: A framework for organizing the planning and implementation of data resources. The set of data, processes, and technologies that an enterprise has selected for the creation and operation of information systems. The blueprint that describes the environment that the data warehouse, analysis application, or operational system is built.

ASP – Application Service Provider: A company whose business is providing application services for its client companies. Such applications can include both tactical systems, such as billing systems, or strategic solutions such as CRM. (CRM ASPs currently account for over half the ASP market.)

Atomic data: Data at its most granular and detailed level.

Attributes: In logical data modeling, attributes of an entity refer to the properties of that entity. Each property will have one distinct value per instance of the entity. Example: Entity = Automobile, Attribute = color, attribute value = red. When logical models are translated into physical data models, entities become tables, and attributes become columns. Note: there is not necessarily a 1:1 correlation between the logical model objects and the physical model objects.

Availability: The percentage of time during scheduled hours that the system can be used. It also can refer to the days/week and the hours/day that the system is scheduled for use. See Service Level Agreements

Back-end: Populating the data warehouse with data from operational source systems.

Base table: In relational databases, tables are defined as temporary or base. Base tables are the tables that are created by the CREATE TABLE command and are used for persistent storage.

Batch windows: The time that is required to run the ETL process from beginning to end.

Best-of-breed: Refers to the most effective, powerful, functional and optimal choice of product in each category of tool. As organizations choose tools, they must decide whether they wish to chose a suite of products from the same vendor (where some of the tools in the suite are not terrific) or choose the best product in each category, i.e., best of breed, and integrate those tools themselves.

Best practices: Processes and activities that have been shown in practice to be the most effective.

Beta release: A version of the vendor's software that is given to selected installations prior to the product becoming generally available. This version is often not free of defects.

Big Bang (approach): Delivering all the intended functions of the data warehouse at the same time.

Bitmapped indexes: This is an alternate (to B-tree) indexing mechanism that involves building streams of bits where each bit is related to the column value for a single row of data in a table. The use of bitmapped indexes on low-cardinality fields (fields that have few possible distinct values) improves query performance significantly.

Boilerplate: Standard verbiage that can be used multiple times for the same purpose. Vendors respond to RFPs with boilerplate so they do not have to write the same material multiple times.

Business analyst: The person whose job it is to analyze the operation and data of the business to develop a business solution.

Business drivers: The tasks, the information and the people that promote and support the goals of the enterprise. The requirements that describe what the business wants (e.g., more quality data, faster response to queries). A problem in the business that is important enough to spell the difference between success and failure for an organization.

Business intelligence (BI): Normally describes the result of in-depth analysis of detailed business data. Includes database and application technologies, as well as analysis practices. Sometimes used synonymously with "decision support," though business intelligence is technically much broader, potentially encompassing knowledge management, enterprise resource planning, and data mining, among other practices.
Business process engineering: The analysis and re-design of business processes and associated technology systems, with the goal to eliminate or reduce redundancy and streamline interactions.

Business rules: Policies by which a business is run. The business rules contain constraints on the behavior of the business. The assertions that define data (e.g., the state code business rule might be the 50 United States, the District of Columbia and the U.S. Territories) from a business point of view.

Business sponsor: Manger or executive who acts as visionary for the data warehouse program and can articulate how the data warehouse can drive business improvements. Establishes the "need, pain, or problem" the data warehouse will solve, serves as a tiebreaker for issues during the project, and might actually fund some or all of the data warehouse development.
See Sponsor

Business timestamp: A business timestamp is a timestamp that is generated by a business event and not a result of a systems operation. Examples are: sales_timestamp, order_timestamp, shipment_date, etc. Typically, all facts in a Data Warehouse have at least one business timestamp, which can be traced to a transaction in the source operational system.

Business users: Personnel reporting to the line-of-business who access the

data warehouse by writing reports and queries or who use the reports and queries generated by others.
See Users

Caching: As related to caching reports, this involves storing the results of pre-run reports in tables (instead of caching to memory as the usage of the word implies) so that when the user accesses the report for the first time, it seems to run instantaneously. This is a feature provided by the server component of many of the popular OLAP tools.

Campaign Analysis: Campaign analysis provides a measurement of responsiveness to campaigns by households and by individual customers. It provides the ability to measure the effectiveness of individual campaigns and different media and offers the ability to conduct cost-benefit analysis of campaigns.

CEO: Chief Executive Officer

CFO: Chief Financial Officer

Champion: The (high level) person in the organization who supports and promotes the data warehouse, its use, and those who developed and maintain it. A person with sufficient clout in the organization who believes in and sells the idea of the data warehouse and helps solve problems between groups.

Channels: The method/means by which a product or service is marketed, ordered, and delivered.

Charge back: The process of assessing and assigning the costs of a system to the departments that use it.

Check totals: "Check totals" is a loose term used to describe the total sum of the values in an additive column of data across all rows of data that are within scope. This total is usually calculated before and after moving data across platforms or processing data in order to ensure no data was lost.

CIO: Chief Information Officer

Class: A collection of objects that share common properties, common definitions and common behaviors.

Clickstream: Series of page visits and associated clicks executed by a Web site visitor when navigating through the site. Analysis of clickstream data can help a company understand which products, Web site content, or screens were of most interest to a given customer.

CMM: Capability Maturity Model: Developed by the Software Engineering Institute (SEI), the CMM is a representation of the goals, methods, and practices needed for the industrial practice of software engineering. The goal of the model is to have processes that are repeatable, defined, managed, and optimized.

Conformed dimensions: A dimension defines the organization of the measures (facts), or it is an entity "how" an organization measures a fact. A conformed dimension is a dimension that is agreed upon its use and semantics across the enterprise, which makes it "conformed".

COO – Chief Operating Officer

Consultant: A consultant is someone who provides expertise and can be an advisor or a deliverer of tasks. Consultants are hired for their expertise when the company has none
Consultants often help define the data warehouse strategy and assess the organization's ability to implement the data warehouse.

Contractor: A contractor is a person who provides the delivery of tasks. The contractor might be responsible for building the ETL process or for overseeing the DBA functions. Contractors are hired when the company has a shortage of skilled workers. The company tells them what needs to be done, and the contractors perform the work.

Control totals: The addition of values of specific fields to verify that the ETL job streams have executed properly. Cross footing of numbers to verify that a process (e.g. ETL) has executed successfully.

Corporate information factory (CIF): The framework that exists that surrounds the data warehouse; typically contains an ODS, a data warehouse, data marts, DSS applications, exploration warehouses, data mining warehouses, alternate storage, and so forth.

Cost/benefit analysis: The process by which the value of a project is estimated based on the expected costs compared to the tangible benefits usually expressed as increased revenue, or reduced cost.

Critical success factor: An element that contributes to the success of a project, without which the project will fail.

CRM – Customer relationship management: Infrastructure that enables delineation of and increase in customer value and the correct means by which to increase customer value and motivate valuable customers to remain loyal – indeed, to buy again. A collection of integrated applications, which facilitate the seamless coordination between the back office systems, the front office systems, and the web. The DSS expansion of CRM Analytics refers to customer-centric analytics applications.

Cross organizational: Includes multiple departments within an organization. A non-redundant and horizontally cross-functional view of the business.

Cross Selling: Selling an additional category of products as a result of the customer's original purchase.

CTO: Chief Technology Officer

Customer Segmentation: Separating customers by factors such as age, gender, educational background, and liking or disliking Wayne Newton.

DA – Data administrator: The role responsible for the enterprise's data resources and for the administration, control, and coordination of all data related analysis activities. The DA has the responsibility for planning and defining the conceptual framework for the overall data environment. The functions of the DA typically include requirements definition, logical data modeling, data definitions, logical to physical mapping, maintenance of inventory of the current system, data analysis, and the meta data repository.

DASD: Rotating magnetic disk storage.

Data architecture: The framework for organizing the planning and implementation of data resources. The set of data, processes, and technologies that an enterprise has selected for the creation and operation of information systems.

Data analysis: The systematic study of data so that its meaning, structure, relationships, origins, etc. are understood.

DBA – Database administrator: The Database Administrator is responsible for the physical aspect of the data warehouse. This includes physical design, performance, and maintenance activities including backup and recovery

Data loading: The process of populating a data warehouse. It may be accomplished by utilities, user-written programs, or specialized software from independent vendors.

Data mapping: The process of identifying a source data element for each data element in the target environment.

Data mart: An implementation of an analytics application serving a single department, subject area, or limited part of the organization. Usually refers to a physical platform on which summarized data is stored for decision support. Data marts are commonly used for specific analysis purposes by a single organization or user group.

Data mining: Discovery mode of data analysis, or analyzing detail data to unearth unsuspected or unknown relationships, patterns and associations that might be of value to the organization. Advanced analysis used to determine certain patterns within data. Most often associated with predictive analysis. A process of analyzing large amounts of data to identify patterns, trends, activities, and content of data content relationships.

Data ownership: Responsibility for determining the required quality of the data, for establishing security and privacy for the data and determining the

availability and performance requirements for the data. Data originators who have the authority, accountability, and responsibility to create and enforce organizational rules and policies for business data.
See Ownership

Data stewardship: Responsibility for the quality of the business data; an information expert about a particular subject area.
See Stewardship

Data Warehouse Manager: The data warehouse has overall responsibility for all the organization's data warehouse initiatives, for data warehouse standards, and for data warehouse tools. The data warehouse project managers may report to the data warehouse manager or they may report to individual sponsors.

Data Warehouse Project Manager: See Project Manager

Data quality: The degree of excellence of data. Factors contributing to data quality include: the data is stored according to their data types, the data is consistent, the data is not redundant, the data follows business rules, the data corresponds to established domains, the data is timely, the data is well understood, the data satisfy the needs of the business, the user is satisfied with the validity of the data and the information derived from that data, the data is complete, and there are no duplicate records. For example, this means that a customer's name is spelled correctly and the address is correct.

Data staging: The storage of data prior to it being loaded into a data warehouse or data mart.
See Staging area

Data Warehouse: A collection of integrated, subject-oriented databases designed to support the DSS function, where each unit of data is relevant to some moment in time. The data warehouse contains atomic data and lightly summarized data.

DDL – Data definition language: The SQL syntax used to define the way the database is physically organized.

Deadline: The point in time by which a project must be completed.

Deliverable: The tangible output from a task or a project, e.g. logical model, project agreement, database design or application.

Delta: A change, e.g. the difference from one period to the next.

Demo: Short for demonstration as in a vendor demonstration of software to impress the users.

Denormalization: Data or data design elements that do not conform to the rules of data normalization. Denormalized data structures are often used in databases to provide rapid access for specific user needs. Denormalization usually results in some degree of data redundancy in a data record. A process of combining like data into a single entity (table or file). This combining will create duplicate data.

Departmental systems: A data mart implementation that serves the needs of only a single department such as Human Resources or Finance
See enterprise systems

Derivations: The transformation of data in the ETL process in which the data is created through the use of an algorithm based upon data from multiple sources.

Derived data: A new data element that is created from or composed of other data elements.

Design review: A peer review of project deliverables, such as design specifications, program code or test specifications. The objective of the review is to find weaknesses, errors and problems. The process where different groups are given access to the design to provide input on how it might be changed to 1) work best with the tools selected or 2) be complete in its solving the problem.

Dimensional hierarchy: A dimensional hierarchy refers to the different levels of data within a dimension that data can be rolled up to or down to for analysis. This can be represented in a data model by a series of related tables with parent-child relationships (snow-flaked schema's) or by multiple columns within a dimension table (standard star schemas) called hierarchy columns. Example: the dimensional hierarchy of a sales organization could include the following levels: salesperson, branch, territory, region, company.

Dimension data: An entity used to describe, qualify, or otherwise add meaning to "facts" in a star schema fact table. Dimensions are the "by" items in analysis of facts "by" product, market, time, period, etc. Descriptive data that describes the measurements (facts) that business users wish to analyze.

Domain (synonym valid values): A set of data values which represent the full range of allowable values that may be used for a given data attribute. Defines validity criteria for a particular column or field. Domains include data types and valid values. For example, Gender could be a domain defined as have the data type of Character of 1 byte containing "F" for Female, "M" for Male, and "N" for Not so Sure.

DSS: Decision Support System
See Decision Support System

EIS: Executive Information System: A system that lets upper management view the organization's performance at a highly summarized level and usually in a graphical representation.

End-user: See User, Business User

Enterprise data model: A logical data model that incorporates all the important components of an enterprise data architecture. Components include entities, attributes, relationships, rules and definitions stated in business terms. A schematic defining the data and their relationships that is applied to the whole organization. Diagram of a single non-redundant view of business data, showing how data is used by the business activities of an organization.

Enterprise Data Warehouse: A collection of data that can be defined and shared across the whole enterprise along the lines of common dimensions to be used for analysis.

Enterprise systems: Systems that support and are used by the entire enterprise
See departmental systems

Entity: A person, place, thing, concept or even about which an organization collects data.

ERP: Enterprise Resource Planning: Tying together and automating of diverse components of a company's operations, including ordering, fulfillment, staffing, and accounting. This integration is usually done using ERP software tools.

ETL: Extract/Transform/Load: This is the process of extracting data from their operational data sources or external data sources, transforming the data which includes cleansing, aggregation, summarization, integration, as well as basic transformation (1 becomes "Male" 2 becomes "Female"), and loading the data into some form of the data warehouse (ODS, enterprise data warehouse, data mart). ETL can also refer to the vendor software that performs these processes.

FAQs: Frequently Asked Questions: Questions that are repeated, usually asked by the users of the help desk or of the project support team. Software vendors also have FAQs which are usually asked by technical people who support the vendors' software. To minimize support requirements and to assure a consistent response, FAQs are normally captured, validated, and made available through a web site.

Fact table: The central table in a star join schema, characterized by a composite key, each of whose elements is a foreign key drawn from a dimension table. Facts are information about the business, typically numeric and additive. A table that contains the measures that the business users wish to analyze to find new trends or to understand the success or failure of the organization.

Federated database system (FDS): A federated database system is a collection of independently managed, heterogeneous database systems that allow partial and controlled sharing of data without affecting existing applications. An FDS presents an enterprise view of data.

Foreign keys: Foreign keys are columns on one table that are inherited from the primary key of another table by means of a dependent or independent relationship.

Front-end: The access and analysis piece of the data warehouse architecture.

FTE: Full time employee, Full time equivalent

FTP – File transfer program: A program that transfers data from one computer to another.

Gap Analysis: The difference between what is needed and what is available. The difference between where you are and where you want to be.

Global 2000: The 2000 largest companies worldwide.

Goal: An objective to be achieved within a specific period of time.

Granularity: The level of the measures within a fact table represented by the lowest level of the dimensions.

Hard dollar (benefits): Tangible benefits that can be measured. Hard dollar benefits can result from an increase in revenue or a reduction in cost.

Historical data: Data from previous time periods, in contrast to current data. Historical data is used for trend analysis and for comparisons to previous periods.

Infrastructure: The architectural elements, organizational support, corporate standards, methodology, data, processes, and physical hardware/network, etc. that make up the data warehouse environment.

Integration: The activity of combining data from multiple data sources to present a single collection of data to the warehouse.

Islands of automation: Systems that were developed without consideration for their ability to interface with each other. As a result, data stored in these systems is often redundant and inconsistent.
See silos, stovepipes

IPO: Initial public offering

IT – Information Technology: The department that builds and maintains computer systems.

Iteration: The division of a project in which functionality is provided to the users in a series of phases.

Joins: Within the context of SQL, joining refers to the comparison of similarly valued keys across multiple tables for the purpose of selecting rows of data from multiple tables. This is done by means of an SQL SELECT statement where the comparison of the keys is performed in the WHERE clause.

Justification: The process by which each project is evaluated to determine if there is financial viability in its implementation. The justification process also allows management to prioritize projects.
See cost/benefit, ROI

Knowledge Transfer: The act of transferring knowledge from one individual to another by means of mentoring, training, documentation, and other collaboration.

Legacy system: Any existing production or operational system. Legacy systems often provide the source data for the data warehouse.
See Operational Systems

Libraries (queries and reports): Sets of programs that have been created, fully tested, quality assured, documented, and made available to the user community. The programs in these libraries are variously called canned, predefined, parameterized, or skeleton queries/reports. They are launched by the user, who only enters a variable such as a date, region number, range of activity or some other set or sets of values the program needs to generate a query or report.

Line of business: Divisions of a company responsible for the production and creation of the organization's products and/or services. IT, HR and Accounting are not lines of business.

Logical data model: An abstract formal representation of the categories of data and their relationships in the form of a diagram, such as an entity-relationship diagram. A logical data model is process independent, which means that it is fully normalized, and therefore does not represent a process dependent (e.g. access-path) database schema.

Market Penetration: The percentage of the market owned by a company as represented by share of revenue.

Matrix management: A reporting structure in which the manager does not hold the performance and payroll card of the subordinate. This is synonymous with dotted line responsibility.

Mentor: A person who provides guidance and recommendations to a more junior person for courses of action and behavior.

Meta data: "Data about data." Usually refers to agreed-on definitions and business rules stored in a centralized repository so business users – even those across departments and systems – use common terminology for key business terms. Can include information about data's currency, ownership, source system, derivation (e.g. profit = revenues minus costs), or usage rules. Prevents data misinterpretation and poor decision making due to sketchy understanding of the true meaning and use of corporate data.

Methodology: Proven processes followed in planning, defining, analyzing, designing, building, testing, and implementing a system.

Metrics: Any type of measurement. Metrics could include business results, quantification of system usage, average response time, benefits achieved, etc. The measures that an organization believes is vital for its success.

Milestone: A tangible event used to measure the status of the project. Markers during the execution of a project that shows the movement of a project in the right direction.

Mission: A high level set of goals of the organization. For example to be the low cost producer or the company with the highest level of customer satisfaction.

MPP – Massively Parallel Processing: A parallel hardware organization that de-emphasizes the sharing of memory resources.

Multidimensional: The aggregation of data along the lines of the dimensions of the business, e.g. sales by region by product by time.

Near-line storage: Data storage that is not on-line and not with immediate access.

Networking:
(1) Connecting with people of like interests for the purpose of uncovering opportunities, identifying landmines and learning of best practices.

(2) The ability to tie more than one component together through protocols (e.g. TCP/IP)

Object: An instance which is a member of a class.

Objective: Desired outcome of the delivery of the project. An objective can be measured.

OCM: Organizational Change Management

OLAP – Online Analytical Processing: "Drilling down" on various data dimensions to gain a more detailed view of the data. For instance, a user might begin by looking at North American sales and then drill down on regional sales, then sales by state, and then sales by major metro area. Enables a user to view different perspectives of the same data to facilitate decision-making.

OLTP – Online transaction processing: Defines the transaction processing that supports the daily business operations.

OO – Object oriented: A self-contained module of data and its associated code.

Operational data: Data that supports the productions systems that run the business. This includes, but is not limited to, OLTP systems.

Operational system: The system that creates, updates and accesses production systems. They do not access, or update decision support systems. See Legacy system

Organizational change management: Major change is defined as those situations in which performance of job functions require most people throughout the organization to learn new behaviors and skills. Major change encompasses an entire workforce and can focus on innovation and skill development of people.

To some degree, the downside effects of change are inevitable. Whenever groups of people are forced to adjust to shifting conditions, discomfort will occur. The key is to proactively recognize the effects of change, plan for the change, and develop skill sets and tools to support the change and inevitable discomfort associated with it. Without this proactive approach, the risk of poor project implementation increases significantly and reduces the opportunity to achieve expected compliance.

Outsourcing: Assigning responsibility for all or a portion of the activity and tasks involved in developing and/or running and maintaining a system to a vendor outside of the organization.

Ownership, Owners of source data: One of the more controversial and disputed ideas. The person or group who has responsibility for determining who can access the data warehouse (security), the domains of the data, the performance and availability requirements.
See Data Ownership

Pain: An unfulfilled business need that jeopardizes the success of the organization.

Parallelism: The ability to run the same process simultaneously (in parallel) within more than one processors.

Partitioning: The ability to divide a table into pieces (partitions). The division can be horizontal (by data value – for example by date) or vertical (by columns – for example, most used columns in one partition, the least used columns in another partition.).

Periodicity: The frequency of load/update/refresh of the data warehouse, e.g. daily, weekly, monthly.

PERT Chart: A graphical representation showing the critical path for a project applied to a calendar.

Phasing: The method of delivering the data warehouse in separate groupings of functionality to particular groups of users rather than delivering everything all at once to all the intended users.

Physical Data Model: A formal representation of data and their relationships in the form of a diagram, depicting the physical placement of data in a database. A physical data model is process dependent, which means that it is denormalized to provide maximum performance efficiency. It is commonly referred to as logical database design or database design schema.

Pilot: The initial implementation of a data warehouse. A pilot is always a subset of the intended function and would include a subset of the total set of users. A partially built system to show the capabilities of a full implemented system. A pilot should not become a live system, but usually does. A pilot, proof of concept and prototype are sometimes used synonymously.

Platform: The hardware, operating system and database management or file system
on which the data warehouse runs.

Political agenda: The plans of an individual to enhance his or her position in the organization.

Power users: Knowledge workers who are capable of writing complex queries and reports with little need for help.

Primary key: Refers to the column(s) on a relational table that uniquely define a row of data on that table.

Project agreement: A document outlining the scope of a project including the deliverables, the functions, tools to be used, service level agreements, responsibilities and schedule. The project agreement sometimes includes the anticipated milestones.

Project Manager: Sometimes referred to as the data warehouse project manager, the Project Manager has overall responsibility for a project's successful implementation. The Project Manager defines, plans, schedules, and controls the project. The project plan must include tasks, deliverables and resources – the people who will perform the tasks. The manager will monitor and coordinate the activities of the team, and will review their deliverables. If contractors and consultants are used, the Project Manager assigns the tasks, monitors activities and deliverables and assures that knowledge transfer is indeed taking place.

Project Management Office: Sometimes called project office. This is the office or department responsible for establishing, maintaining and enforcing project management processes, procedures, and standards. It provides services, support, and certification for project managers.

Proof-of-concept: Software trial that allows a prospect to try out the product before buying it. Delivers a realistic slice of functionality and is often used as the foundation for the first application. A quickly built system to show the capabilities of an idea. A proof-of-concept should not become a live system, but usually does. A pilot, proof of concept and prototype are sometimes used synonymously.

Prototype: A less formal experimental and experiential development process of a proposed application for the purpose of demonstrating some or all of its functional capabilities. A prototype does not have the same rigorous testing, documentation, and implementation requirements as a software release or an application does, and should therefore never be implemented as-is.

Quality: The absence of any defect. The characteristics of a system that conforms to the original design. A system of quality would have the following characteristics: 1. Maintainability (easy to add new functions), 2. Conformance to specifications (fulfilling end user requirements), 3. Long mean time to failure (few bugs and abnormal terminations), 4. Performance that is adequate or as expected, 5. Well tested for functionality, user interface, and performance, 6. Well documented, 7. Easy to use, and 8. Uses standard interfaces.

QA - Quality Assurance: The department, role or process responsible for validating that which is proposed to ensure a correct outcome. The planned and systematic activities to provide confidence that a product or service will fulfill requirements for quality.

RAD: Rapid Application Development
A process where the time is set (timeboxed) and a small set of deliverables is implemented in a reasonably short period of time.

RDBMS – Relational database management system: e.g. DB2, Oracle, SQL Server, Sybase

Real time: Data that is captured, and made available as it is happening. Real time data reflects the latest status of the organization's operational transaction data. Current moment in time. Real time refers to what is happening to any piece of data right now. For analysis, some people want to see current rather than historical data as is the case with most data warehouses.

Recursive: A relationship between two instances of the same entity, as in "recursive data design".

Referential integrity: The concept of enforced relationships between tables based on the definition of a primary key and foreign key.

Release concept: A new approach to development that produces a fully tested, fully documented, high-quality, but only partially functioning application until the final release, which completes the application. The release concept severs the notion that a project deliverable must equal a complete application. Instead it tightens and expands on the concept of a pilot by producing a partially functioning application, which is refined and enhanced several more times through several more releases before it becomes a fully functioning application. This concept is the embodiment of iterative development and is fully compatible with XP (extreme programming) and the new agile and adaptive methodologies.

Resources: People and budget needed to perform the data warehouse tasks

RFP - Request for Proposal: A formal request to a vendor to submit a proposal to provide a product or service.

ROI – return on investment: Usually represented as a percentage of tangible monetary value in relation to the cost of the system.

Rolled up: Aggregated to a higher level

Scalable: Ability to increase the number of users, the size of the databases and the complexity of the queries and reports without having to replace the existing platform or architecture.

Scope: An itemized accounting and definition of the agreed upon project deliverable in terms of functionality as well as data. In data warehousing, the data scope is more critical than the functional scope for correctly estimating the development effort.

Scope creep: The addition of new requirements, source data or users to the initial agreement of what the project will be delivering.

Semantic layer: A layer between the end-user tool and the database. This allows the end-user tool to present the data most effectively for the end-user understanding and then to generate the proper query to the database.

Service level agreement (SLA): The definition of a level of service provided by the IT department for a particular system. Service level agreements can be established for availability (24 hours/day, 7 days/week and 98% during scheduled hours), for performance (response time for 95% queries in 1 minute or less), for timeliness of the data (weekly data available 6 AM Monday morning), or for other reasons. Contract with a service provider – be it an internal IT organization, an ASP, or an outsourcer – specifying discrete reliability and availability requirements for a given system. Might also include such requirements as support of certain technology standards or data volumes. Outsourcer's failure to adhere to the terms laid out in the SLA could result in financial penalties.

Sign-off: The process of agreeing – in writing – to the scope of a project or the acceptability of a deliverable.

Silo, siloized: A silo system cannot easily integrate with any other system. This means we have multiple versions of the same data, violating the idea of a single version of the truth.
See stovepipe and Island of automation

Single version of the truth: A primary goal of the data warehouse wherein the data to be accessed resides in only one database so that there will be no conflicting data and no inconsistent reports.

Shelfware: Software that is not being used as in "sitting on the shelf".

SMP – Symmetrical Multi-Processing: A parallel hardware organization that emphasizes the sharing of memory resources.

Snowflake structure: Snowflake is a star schema with normalized dimensions.

Source data: The data from the operational or legacy systems that feed the ETL process.

Source system: An operational system, or ODS that is used as the source or input to the ETL process.

Sponsor: The person in the organization, usually from the business side – who supports the project. This person should be someone with power, money and commitment to the project.
See Business Sponsor

Staging area: A staging area is where the ETL programs execute and where the source data is prepared for the data warehouse.
See Data Staging

Stakeholders: People who have a vested interest in the success of the project or are involved in the implementation of the project.

Standards: A standard is "Thou shall" while a guideline is a recommendation, more like "You should if your situation warrants." Data warehouse standards examples include: meta data, terminology, data stewardship, and privacy.

Star schema: A modeling paradigm that has single object in the middle (fact table) connected to a number of objects (dimensions tables) around it radially.

Stovepipe: A stovepipe system cannot easily integrate with any other system. This means we have multiple versions of the same data, violating the idea of a single version of the truth.
See silo and islands of automation

Strategy: Approach taken that will affect the overall direction of the organization and will establish the organization's future environment.

Subject areas: Data Subject Area: Fundamental entities that make up the major components of the business, e.g. customer, product, employee.

Function Subject Area: A business function or business activity, e.g. sales, order processing, inventory.

Suite (of products): A collection of software products from the same vendor – either developed or bundled by that vendor. The idea is to provide a complete set of tools from modeling through to access and analysis. Range of functional software modules that interact with each other. Suites should eliminate integration complexity.

Supply chain: The management of the components, manufacturing and distribution of a manufactured commodity. The supply chain management includes warehousing and tracking inventory.

Systems integration: The art and science of integrating processes, functions, people and data so the end result is a seamless and tight knit system.

System timestamp: A system timestamp is a timestamp that is generated by a systems operation. Examples are : record_create_date, last_update_date,...

SWAT team: A small team of skilled and experienced practitioners who can pull a failing project out of the ditch. This team does not tolerate political interference as it makes decisions and takes actions to bring the project to fruition.

Tactical: Approach taken to achieve a specific objectives or to solve a specific problem.

Target: The database into which data will be loaded from a source database or file; the data store that is accessed by the users.

Terabyte: 1000 Gigabytes.

Third normal form: A database in which each attribute in the relationship is a fact about a key, the whole key and nothing but the key. Usually refers to a fully normalized structure.

Tie (and foot): The process of validating the number of rows, summarizations, and monetary totals of the source data to the data loaded into the data warehouse.

Timely: Data is valuable and useful to analysts only if it represents organizational activities that are reasonably current. Timeliness is a function of the users' requirements for currency and is consistent with user expectations. Timeliness is usually measured by how soon the data is available after some distinctive end-of-period such as "two days after the close of the month." The act of getting the data to the users at the most opportune time.

Time dimension: A table of descriptive attributes about the date/timestamp, e.g. Day of week, Month, Quarter, Season, Year, Century, Holiday, etc.

Time variance: A characteristic of a data warehouse that defines the moment in time that the data or variant of the data is valid. If Order No. 123 has a value of $1,500.00 on Dec 1 and $1,700 on Dec 10, Dec 1 and Dec 10 shows us the time variance of Order No. 123.

Topology: The manner in which the components of a subject are arranged or interrelated.

Total Cost of Ownership: The cost to the organization for the initial implementation and the maintenance of the system.

Transformation: The manipulation of data to bring it into conformance with the business rules, domain rules, integrity rules, and with other data within the warehouse environment.

Triage: The process by which projects or activities are prioritized to determine which should be attempted first, second, etc. and which projects or activities should never be done at all. This process applies to the cleansing process to determine which data should be cleaned first, second, etc. and which data should not be cleaned at all. Triage considers the value of cleansing, the complexity and the cost and the order in which the cleansing should be accomplished.

Trickle feed: The process by which data updates the target database a little at a time. This is in contrast to massive updates that take place after the close of a period such as the day, month or quarter. The process of feeding data from one system to another in either real-time or small time intervals.

UPC – Universal product code: A unique bar code embossed on every product used for inventory control.

User: A knowledge worker, a business analyst, a statistician, or a business executive who will access the data in the data warehouse to perform some type of business analysis.
See Business Users

Value added: The notion of additional benefit being provided by some activity or service.

Virtual enterprise data warehouse: An enterprise data warehouse constructed of multiple data marts and a request broker computer application. The data warehouse does not physically exist except through out the formation of the integrated data marts.

Vision: The direction of the data warehouse – what it is intended to accomplish.

Visionary: The person in the organization who articulates the data warehouse direction – what it is intended to accomplish.

Visualization: The presentation of results in a format other than just numbers with a display that may include graphs, and charts making copious use of colors and figures.

VLDB (Very Large Database): The perception of what constitutes a VLDB continues to grow. A one terabyte database would normally be considered to be a VLDB.

Work Breakdown Structure: A detailed list of tasks to be performed on the project.

Workload: The quantity of processing to include the machine cycles and the disk I/Os.

Stars: A Pattern Language for Query Optimized Schema

For years organizations have deployed and used large on-line transaction processing(OLTP) systems to automate and record their business activity. The challenge now is to allow business analysts, those individuals in the organization chartered to support decision makers by producing reports, to access this data in an ad hoc manner. While OLTP schema are optimized for data entry, they seldom provide an acceptable solution for data analysis.

The star schema concept presented here is the product of many discussions over the years between the author and consultants specializing in Decision Support Systems(DSS). The star schema has also been called a star-join schema, data cube, data list, grid file, and multidimensional schema by practitioners in the field. The name star schema comes from the pattern formed by the entities and relationships when represented as an entity-relationship diagram(ERD); Typically there is a business activity in the center of the star surrounded by the people, places, and things that come together to perform this activity. These people, places, and things can be thought of as the points of the star.

The star schema pattern language presented here is an attempt to provide a method for business anaylsts to develop a schema which is easy to query. Seven patterns are presented which are part of a larger pattern language that support the field of DSS, a branch of Business Data Processing Systems. The seven patterns are split into two sections. The first section addresses the issue of finding and organizing the relevant factors of the business that need to be analyzed. The second section deals with implementing these factors into a star schema for a query system.

Analysis Section. To begin, make a list of those things that are meaningful to your situation as you try to discover the entities that model that part of the business for which you are responsible. Your model should contain the names of things that are readily recognized by anyone working in your area. As the model is refined, it should reflect those activities and the things involved in those activities that are directly relevant to trends that your business is influenced by and can influence. The patterns are:

---

1. Query Optimized Database

In the ongoing operation of your business area, be it Sales, Marketing, Manufacturing, Engineering, or whatever, the need arises to analyze your business in order to solve a problem. These problems could be something like the need to correct poor performance or seek new opportunities. Solutions and opportunities are usually found by analyzing the data that is captured as a part of the ongoing activities of the business. This data is usually found in your company's OLTP database(s).

OLTP database schema are usually optimized for the recording of business transactions. The normalization process for OLTP schema design takes descriptions of entities from the business domain and breaks them apart into a number of small tables. These small tables can then be handled by the

DBMS in an efficient manner. Breaking up entities into small tables also reduces the amount of redundant data stored in the database since these each table can be joined with other tables to form more than one business entity. Reducing data redundancy means that the same piece of data will only be stored in the database once. By having the data stored only once, the problem of updating multiple copies disappears when the data changes.

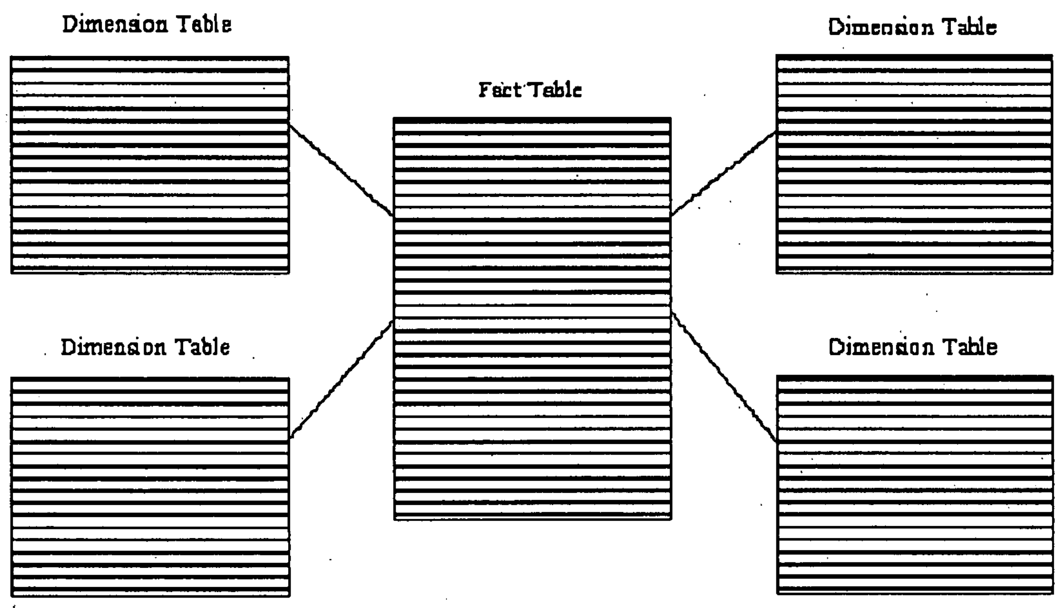
Two problems arise when a you want to query an OLTP schema. First, the fields that describe one of your business entities will be distributed among a number of tables which will then need to be joined together in order to form the things that you will want to use in your analysis. These joins that have to be performed can each require a significant amount of processing by the database system. Second, since these small OLTP tables have to be joined back together to form the original entity, there will usually be multiple combinations of joins from which to choose. These multiple ways of joining tables can lead to varying answers when the tables are used in a query. Even if different combinations of joins happen to yield the same answer, the join process can be error prone to a person who is not knowledgeable about the details of the schema. *Therefore*:

Develop a new database optimized for the purpose of easy query rather than for inserting and updating data. The new database will most likely have to be implemented on a separate machine from the one that hosts the OLTP database since long queries could adversely affect the performance of the data entry process if the two databases are run on the same machine. Besides the performance issue, another reason to separate the two types of databases is that the DSS database will need to have stable data in order to perform meaningful analysis. If the DSS is implemented as views on an OLTP database, the data within the views will be constantly changing as the data entry process captures new and updates old data. The problem with trying to do analysis on constantly changing data is that you cannot hold some of your variables constant while selectively changing one or two other variables, in order to see the effects of these variations on your results. This is called the Twinkling Data Problem.

The new database and machine will incur additional costs for ongoing administration and maintenance. The issue of transferring data between the OLTP and DSS databases will also need to be addressed as the DSS database will need to be refreshed with new data on a periodic basis. Having data from the OLTP system appear on reports generated by the DSS will bring to light the errors that occur during the data capture process. These errors will need to be corrected in the OLTP database by either scanning and cleaning up the data before each transfer to the DSS or, better yet, refining the data capture process to include ways of catching and reducing errors at the time data is entered.

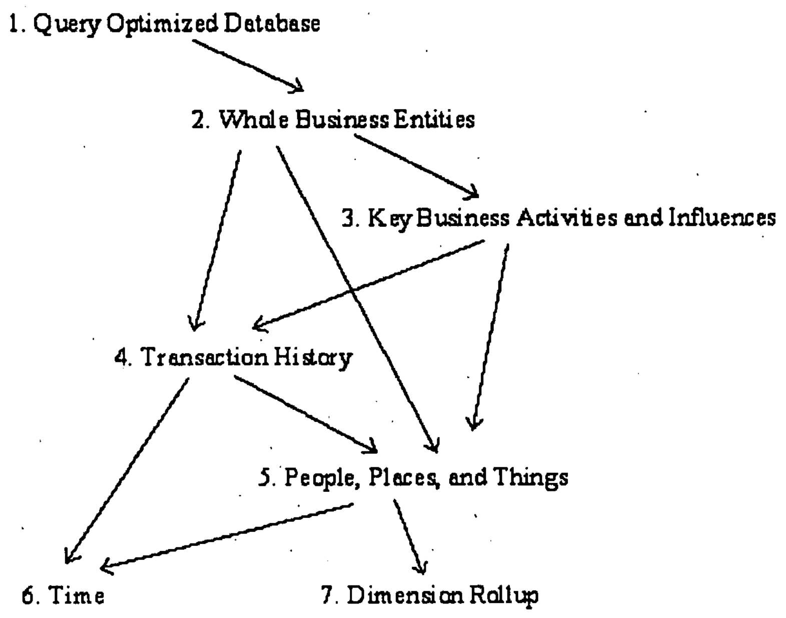
The rest of this pattern language is concerned with finding the entities that you need to model for building a schema which is easy to use. Form Whole Business Entities(2) from your OLTP database that are relevant to your domain. Grouping together all of the relevant data for a given entity will reduce the number of joins in a query. In essence, you will denormalize the OLTP schema to a degree which will allow for faster query. This shouldn't present a problem in the DSS database since it should only be updated on a periodic basis, usually in a batch mode of processing. Using batch processing, the updates to the DSS database can be controlled since they are only coming from one source. This is a much more managable scenario than with an OLTP database where updates are coming from multiple sources in an ad hoc fashion.

Once you have a list of reconstituted entities that model your business area, use Key Business Activities and Influences(3) to focus on those events that are critical to your area of responsibility and the factors which influence them. Its important to make the relationships between the activities and their influencers simple and clear so that forming queries will be easy and accurate. By simple and clear I mean that there is only one join possible between any two tables and the meaning of that join is clear to the person performing the query.



Generic star schema layout

The schema will be developed by defining and implementing two kinds of tables. The first kind of table, called a fact table, will be a Transaction History(4) of the key business activity being modeled. The second kind of table, called a dimension table, will be the People, Places, and Things(5) that are involved in each kind of transaction. Another dimension table, Time(6), is considered separately since it appears frequently in business analysis and deserves special attention. Dimension Rollup(7) is another kind of dimension table which help the person querying the database to specify groups of dimension records


1. Query Optimized Database
2. Whole Business Entities
3. Key Business Activities and Influences
4. Transaction History
5. People, Places, and Things
6. Time
7. Dimension Rollup

Interactions in the Stars pattern language

2. Whole Business Entities

In order to understand a schema that is developed for your business, you will need to use easily recognized names for the objects in your Query Optimized Database(1). Names for the objects in your schema should reflect those things in your environment that you deal with in your day to day endeavors. Each object should be whole in the sense that all of the things that uniquely describe it should be found in one place.

The business entities that you find on the reports of your business activities are usually scattered into a number of small tables in the OLTP database because of the need for normalization. The names for these small tables are usually concatenations of abbreviations for the business functions in which they participate. Your job in this case is that of solving a jigsaw puzzle whenever you want to form a query. Unfortunately for you the same piece can be used in more than one place and you only get one copy. Typically you'll need to ask for help from someone intimately familiar with the details of the OLTP schema. This can be a formidable bottleneck to getting timely answers to your business questions. *Therefore*:

Find the entities in your domain that are directly relevant to your problem by examining the reports you use to monitor your business. The report titles and column headings are the best places to focus on. As you find these entities, make a list of them. Put a short definition next to each entry. Another place for finding entities is from entity-relationship(ER) diagrams of the original analysis done for the OLTP database before it was normalized. The database administrator(DBA) for your OLTP database may have these diagrams available.

```
Example:  List of entities in a sales model
             Customer                 - the people and companies we sell to
             Product                  - the things we sell
             Sales                    - a transaction between our company and a cu
             Sales planning           - setting sales goals and procedures for the
             Sales plan               - a statement of goals for sales and the ste
             Date of sale             - the date of a sales transaction
             Distribution Channel     - the means by which the product was sold to
             Salesperson              - the person who sold the product to the cus
             Competitor               - the other companies who offer competing pr
             Sales Office             - a location where a group of salespeople ar
```

The entities you find here will need to be made whole again from the pieces found in the OLTP database. The fact tables and dimension tables will be where you group together the information for each entity. At this point though, not all of the entries in your list will become tables. You will need to hone in on the things that are important to you by finding the Key Business Activities and Influences(3) that you need to analyze. At this point, don't worry that you have included too many entities; Its more important to brainstorm and find as many as you can. Entities that may seem superfluous at first may lead you to find other useful ones.

3. Key Business Activities and Influences

The things that you find in your business all have a role in the processes that produce your product. Finding the role of each thing is as important as finding the thing itself. Starting with Whole Business Entities(2), you will characterize these objects according to how they are related to one another. By defining these relationships you are also defining the role of each thing. The two categories of things that you will be concerned with will be the activities for which your department is responsibile and the various factors which influence them.

Tables in an OLTP database can have circular references. This usually occurs because normalized tables participate in a number of relationships since the data in the table is being reused. This reuse occurs to reduce data redundantcy. What this means is that I can start specifying the joins between some number of tables that circularly refer to one another in more than one way. This can lead to problems as the order in which I join the tables in one query can yield an answer that is different from a query using the same tables joined in another order. Depeding on the question I'm trying to answer, each join may give an answer that might be right or wrong. Without knowing the meaning of each join beforehand, I can't reliably form a query that will give me the answer I expect. Unfortunately neither the database nor the query language can help me with validating the joins against my intentions since the meaning of the relationships between tables isn't stored anywhere. *Therefore*:

Determine the key business activities in your domain and find the people, places, and things (dimensions) that play some part in one or more of the activities. People, places, and things will be found in the transactions that are recorded for your key business activities. Each transaction will usually describe who, what, when, and where the event took place. These are the simple relationships you need to define between each dimension and its activity. Be careful that the dimensions that you choose should only be related to each other through the key business activity and not with each other. There should be no direct relationships between dimensions except in the case of a Dimension Rollup(7).

In all cases, circular relationships should not be found between the dimensions, the facts, or any combination. It is important to draw out these relationships and make them clear so that you will be able to see the effect that any one of these dimensions has on your business activity. Your job as an analyst will be to see how a change in the value of one of these dimensions influences your key business activities. Starting from the opposite end, once you spot a trend in a business activity you'll want to identify the changes in value of your dimensions to find correlations between the activity and dimensions. Choose the key business activities for your area of responsibility from Whole Business Entities(2). The short definitions will help you spot these. Although these activities should already be found on your list, you may need to look deeper into your business environment to come up with all of the activities you'll need to consider. Limit the activities to only those that are directly related to the mission of your department. Some of the activities on your list may be supporting processes for main activities. Focus on the main activities since these will be critical to your businesses survival and growth. Each key business activity will become a fact table to hold Transaction History(4) in the query optimized database you will build.

For each of the main activities that you identify, make a list of the people, places, and things that are related to this activity. Again, you will probably find these from Whole Business Entities(2), but some may not be present and will only be found through further analysis of your business. Also, be sure to look at the short definition for each item in the list as this will usually have descriptive words which will give important clues for finding these things. Each of the People, Places, and Things(5) will become a dimension table in the query optimized database. The following example assumes that Sales is the key business activity that you should focus on from the example list in Whole Business Entities(2).

```
Example:  List of people, places, and things that are involved in Sales
        Customer

        Product
        Salesperson
```

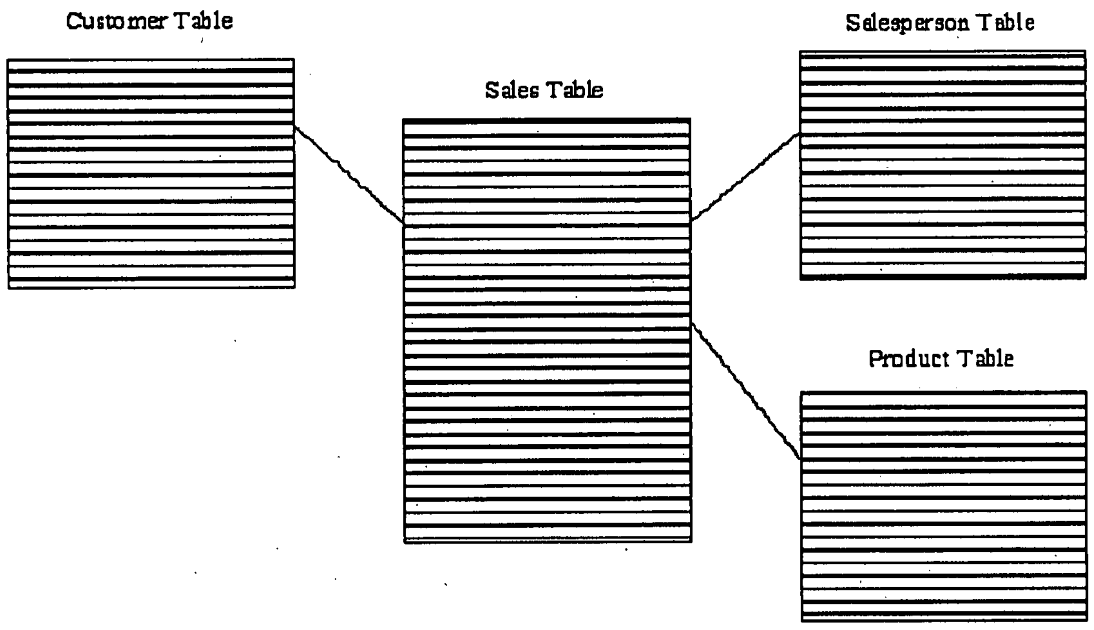
Implementation Section. In order to start analyzing your business model, each entity needs to be represented in your query system. Each entity should take the form of a table or the field of a table, as is appropriate for your particular query system. Each table should be complete in that when a report is created, all of the fields that either need to appear on the report or will be needed to specify the report should be available. Tables will represent the following:

4. Transaction History
5. People, Places, and Things
6. Time
7. Dimension Rollup

---

4. Transaction History

Some activities will be critical to the life of your business. These activities, along with the factors that influence them, are your Key Business Activities and Influences(3). For an activity, it will be convenient to have one place where you can start to refer to all of its dimensions. Here is where you need to build a fact table to contain these references. This will be the center of your star. In addition, the transactions that represent an activity should capture the units of work that are the basis for measuring your business' output. These are the facts about the transaction.



An outline of the Sales star schema

In an OLTP schema, the facts that represent your key business activities, and their dimensions, may be scattered throughout the database because of the normaliztion process. Chains of tables may have to be joined together in order to find all relevant facts and dimensions. This chain may include tables that are irrelevant to your business question but may be needed simply because they contain the references to other tables that you need in your query. This can be wasteful in terms of performing expensive join processing when producing your Whole Business Entities(2). In addition, the join process itself can be error prone as discussed in Key Business Activities and Influences(3). *Therefore*:

Sales Table

Customer
Salesperson
Date of Sale
Product
Quantity Sold
Total Amount($)
Discount

Example of the Sales Table Layout

Construct a fact table that contains the transaction history associated with each activity being modeled. The fact table should have an ID field for each dimension represented as People, Places, and Things(5) related to the transaction. Also put the numeric data, or facts, that are unique to the transaction in the fact table. The fact table is the agent that binds together all of the pertinent facts and dimensions for a transaction. It describes the who, what, when, and where by pointing to the dimension tables and it also contains the amounts of the things that changed hands between the parties of the transaction. Note that its important at this point to confirm with a person knowledgeable in the details of the OLTP schema, typically the DBA, that the data needed to build the fact and dimension tables exists in the corporate database(s).

The facts are usually numeric quantities that describe how much product has been sold and the money received for the product. This numeric data can be summed when a group of fact records are selected. This sum would represent the total activity for some set of criteria specified in the query. Data that is descriptive in nature, like textual names and enumerated type codes, should be placed in the appropriate dimension table.

Sometimes there will be descriptive data that isn't related to any of the dimensions and yet is associated with the transaction. This hard to place data will usually be put in the fact table when there are not enough fields to form another dimension table. In some instances your facts won't be numeric values but instead will consist of only coded fields which capture one of a predefined set of values. In programming terms this would be a field containing an enumerated type. For instance, your fact table would contain mostly enumerated types if you were storing the results of an opinion poll.

The IDs can be thought of being used in two ways. An ID in the fact table can be used to retrieve descriptive information from the dimension table. This is useful when you want to specify on a report the who, what, where or when of a transaction. Another use of an ID field is to find out what has taken place for a combination of people, places, or things. This is accomplished by taking the ID values from each of dimension tables and matching each combination against those stored in the records of the fact table.

The relationship between the records in a dimension table to the records in the fact table will be one-to-many; One record from the dimension table can be related to many records in the fact table. The opposite will usually not be true since you will want to identify only a single person, place, or thing involved in a transaction most time to simplify your analysis. In the case of Time as a dimension, a transaction usually only occurs at one particular time.

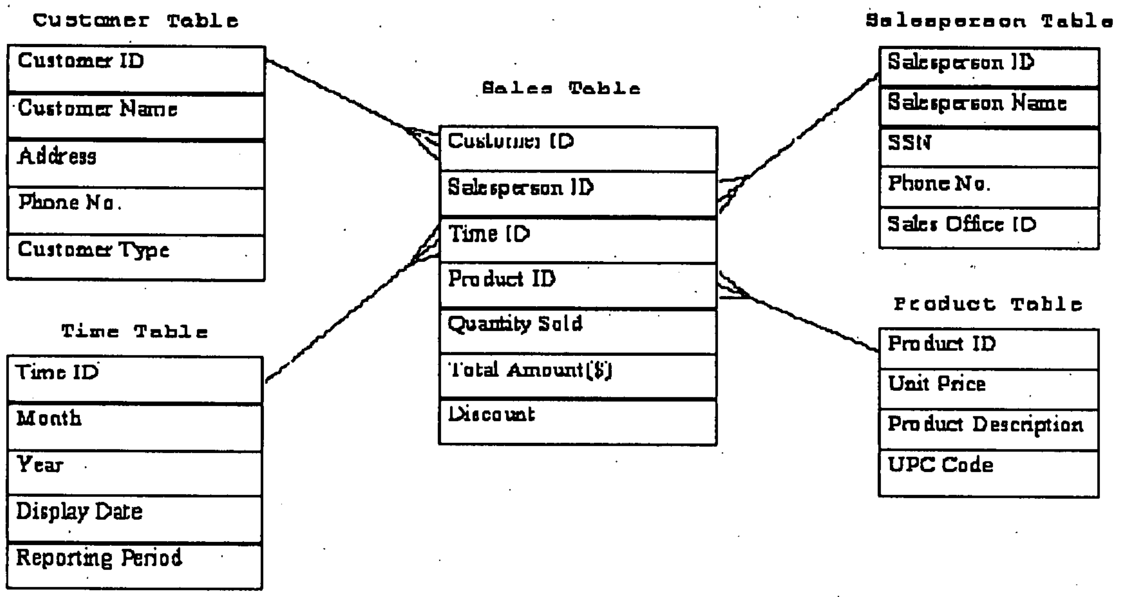
When transferring data from your OLTP databases to your query optimized database(DSS), you will need to keep in mind that you will probably be aggregating data from the OLTP databases. The OLTP database will have data that is captured at the time that the transaction occurs. If the transactions occur daily, then the OLTP data will be stored as of the day the transaction occurred. If your timeframe(Time (6)) for storing data in the query optimized database covers a longer period, then you will need to aggregate the OLTP data while you are making your periodic batch transfers.

Here's an example of the need for aggregation. If you were storing monthly transaction records for sales in your DSS, and you've determined that the dimensions for sales are customer, salesperson, and product, then you will need to aggregate the daily OLTP records into a monthly record for each combination of customer, salesperson, and product that made a transaction. The total number of records possibly created for this fact table would be calculated by multiplying the total number of records for each of the time unit(month), customer, salesperson, and product dimensions together; 12 months X 10 customers X 20 salespeople X 5 products = 12,000 possible records. Since not all of the possible combinations will typically be true for your business in every month, there would only be a fraction of that number of records created.

5. People, Places, and Things

An important part of analyzing your business will be in detecting trends in Key Business Activities and Influences(3). In order to understand the cause of a trend, you will have to look at the different dimensions of the activity and discover correlations between the data values in the dimension tables and the fact table. Here is where you will create those dimension tables that describe the people, places, and things that each record in the fact table will reference. The dimension tables are the points of the star.

In an OLTP schema, the information describing a dimension is usually scattered throughout some number of tables. This can make the generation of reports a challenge as the information needed for the report must be constructed from joining together those tables that contain the fields needed. Besides the complexity of properly determining the joins, the process of joining itself is computationally expensive. Without the capability of being able to easily query your data in an ad hoc manner with reasonable response time, the process of analyzing your business data may be prohibitive. *Therefore*:



Example of the Sales star schema

Develop a table for each person, place, or thing that has a part in the transactions that you are examining. These Dimension tables should model a person, place, or thing completely in so far as to contain all of the fields you need for your queries. You will find these dimensions among your Whole Business Entities(2). The fields will hold quantitative and qualitative values that will be used for displaying information on reports as well as being used for filtering records for calculations. Consider Time(6) as another dimension. Refer to the Sales star schema to see examples of dimension tables.

Since each record in a dimension table will represent a real thing in your business you should have an ID field that contains a value which uniquely identifies each record. The ID field will be used in the fact table as well so that records between the two tables can be joined. Therefore thinking about querying a database with a star schema entails starting from the points of the star, the dimension tables, and specifying criteria for each dimension table which will select some set of records from the center of the star, the fact table. Following is an example which uses the Sales star schema example shown previously.

```
SELECT   c.Customer_ID, Customer_Name, Address, sum(Quantity_Sold)
FROM             Sales s, Time t, Product p, Customer c
WHERE            Month = 3
AND              Year = 1993
AND              UPC_Code = 12678754390                        -- Gum Balls
AND              s.Time_ID = t.Time_ID
AND              s.Customer_ID = c.Customer_ID
AND              s.Product_ID = p.Product_ID
GROUP BY         c.Customer_ID
ORDER BY         c.Customer_ID;
```

This example finds the name and address of each customer who bought gum balls along with the total quantity purchased in the month of March 1993 using SQL syntax. Implicit in the example is the fact that all of the dimensions are related to each other only through the sales fact table. Thus forming queries using this type of schema consists of having a set of simple one-way joins, from each dimension to the fact table. Since no circular joins between tables are possible, at least if you designed your schema correctly, there is no ambiguity of where to start queries. Only in the cases where you have a Dimension Rollup(7) will there be a need to directly relate the dimension tables to one another without relating both dimension tables to the fact table first.

In general when you are analyzing a trend in one of your fact tables, start your query by specifying values for some of your dimension tables, in other words constraining them, and look for correlations in the data values between the uncontrained dimensions and the fact table. For example, if you wanted to find out why a particular product wasn't selling well in a given sales office, you could start by holding the product and time constant and list the types of distribution channels available for all sales offices. It could turn out that the distribution channel mix in the poorly performing sales office is significantly different than in the others, giving you a starting point for further investigation. This process of investigation is called "drilling down" into the data.

6. Time

In order to measure the changes in your business' activity, data has to be analyzed from one particular time to another. These changes may show a trend over a series of subsequent time periods. The trends will be places where you will want to look for causal relationships between your activities and their dimensions.

There is usually no inherent time period that the transactions in an OLTP database are grouped by since all that is needed is a running log of activity from which to guide a continuous process. On the other hand, when you're analyzing one of your business activities, its best to have groups of transactions in order to get a "feel" for what's going on without getting bogged down in the details. Another problem you will encounter when analyzing OLTP data is that time can vary between transactions since they are entered in an ad hoc manner. Yet, the units of time you chose to group transactions should be uniform so that direct comparisons can be made between time periods. These uniform units of time should also not be chosen arbitrarily but should be meaningful to your analysis. *Therefore*:

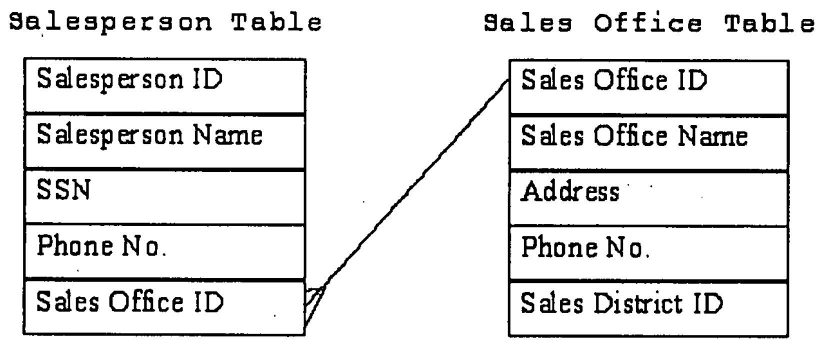
Build a dimension table that will contain units of time that correspond to some significant event in your business. These significant events will usually be the reporting periods when management expects a summary of the business' activity. The unit of time chosen will determine the level at which your transaction data will be aggregated in your Transaction History(4). For example, if you chose your unit of time to be a month, since reports are made to upper management monthly, then daily transactions from the OLTP database will need to be aggregated into monthly transactions. For efficiency, when the data is loaded into your fact table it should be ordered by month so that it doesn't have to be sorted for each query.

7. Dimension Rollup

There will be times when you'll want to "roll up", or aggregate, the level of activity in your business so that you can see the "big picture" for some group of records found in one of your dimension tables. This higher level of aggregation is a convenient technique used in a query. The convenience comes from only having to specify one record in the rollup table to represent many records in the associated dimension table.

The Transaction History(4) will already have been modeled with some level of aggregation assumed. These assumptions for aggregation are based on some intuition about how the data will normally be used. For instance, if you are mainly concerned with tracking the performance of individual salespeople, your fact table will be defined for transactions that reflect the sales of individual salespeople. If the requirement arises that you will need to analyze sales performance on the basis of a geographic area, like a sales office, then all of the sales for the salespeople that are a part of that sales office will need to be analyzed together. Specifying each salesperson that is part of a sales office in a query can be laborious. Alternatively, designating a field of the salesperson table to contain a coded value signifying a particular sales office may be inadequate since there may be other information about the sales office which will need to appear on a report. *Therefore*:

Design a table for the new entity that represents the larger organization that will encompass your existing business dimension. As with People, Places, and Things(5), this new table should include all of the information related to the thing that you will need for including on a report or using as filtering-criteria in a query. You may find that these kinds of things are already in a list of your Whole Business Entities(2).



Example of the Sales Office Table

Like all dimension tables found in People, Places, and Things(5), the table should have an ID field which will uniquely identify each record. This ID field will also be found in the dimension table for which it is aggregating. By having the two tables related by a common ID field, you will be able to easily specify a groups of records in the lower level dimension table by specifying the appropriate value for the ID field in the dimension rollup table. Each of these lower level dimension records will in turn be related to their associated records in the fact table, since these too are related by a common ID field. The fact records can then be summed to yield a quantity that relates back to the rollup dimension. Note that rollup dimensions can be nested to an arbitrary depth to reflect your business. An example of using the Sales Office table to find out the total dollar amount of gum balls that the Portland sales office sold in the month of March 1993 using SQL syntax would be:

```
SELECT   Sales_Office_Name, Product_Description, sum(Total_Amount)
FROM             Sales s, Time t, Product p, Salesperson sp, Sales_Office so
WHERE            Month = 3
AND              Year = 1993
AND              UPC_Code = 12678754390                        -- Gum Balls
AND              Sales_Office_Name = 'Portland'
AND              s.Time_ID = t.Time_ID
AND              s.Salesperson_ID = sp.Salesperson_ID
AND              sp.Sales_Office_ID = so.Sales_Office_ID
AND              s.Product_ID = p.Product_ID
GROUP BY         Sales_Office_Name;
```
Proceedings of PLoP'94, Monticello, IL, August 1994

Author: Stephen Peterson
Sequent Computer Systems, Inc.
stevep@sequent.com

22

Multidimensional Data Modeling for Complex Data

Torben Bach Pedersen and Christian S. Jensen

November 13, 1998

TR-37

A TIMECENTER Technical Report

Title Multidimensional Data Modeling for Complex Data

Copyright © 1998 Torben Bach Pedersen and Christian S. Jensen. All rights reserved.

Author(s) Torben Bach Pedersen and Christian S. Jensen

Publication History March 1999. In Proceedings of ICDE '99
November 1998. A TIMECENTER Technical Report.

Abstract

Systems for On-Line Analytical Processing (OLAP) considerably ease the process of analyzing business data and have become widely used in industry. OLAP systems primarily employ multidimensional data models to structure their data. However, current multidimensional data models fall short in their ability to model the complex data found in some real-world application domains. The paper presents nine requirements to multidimensional data models, each of which is exemplified by a real-world, clinical case study. A survey of the existing models reveals that the requirements not currently met include support for many-to-many relationships between facts and dimensions, built-in support for handling change and time, and support for uncertainty as well as different levels of granularity in the data. The paper defines an extended multidimensional data model, which addresses all nine requirements. Along with the model, we present an associated algebra, and outline how to implement the model using relational databases.

1 Introduction

With the continued advances in the underlying hardware technologies for on-line mass storage and the recent focus on data warehousing, the notion of On-Line Analytical Processing (OLAP) [5] is attracting increasing interest, as business managers attempt to extract useful information from large on-line databases in order to make informed management decisions.

Reports indicate that traditional data models, such as the ER model [2] and the relational model, do not provide good support for OLAP applications. As a result, new data models based on a *multidimensional* view of data have emerged. These multidimensional data models typically categorize data as being *measurable business facts* (measures) or *dimensions*, which are mostly textual and characterize the facts. For example, in a retail business, *products* are sold to *customers* at certain *times* in certain *amounts* at certain *prices*. A typical fact would be a *purchase*, with the amount and price as the measures, and the customer purchasing the product, the product being purchased, and the time of purchase as the dimensions.

In OLAP research, most work has concentrated on performance issues; and higher-level issues, such as conceptual modeling, have received less attention. Several researchers have pointed to this lack in OLAP research, and it has been suggested to try to combine the traditional OLAP virtues of performance with the more advanced data model concepts from the field of *scientific and statistical databases* [9]. This appears to be a very valuable direction, as it is necessary to put more semantics into the database schema to support the typical OLAP style of working directly with the data instead of using pre-formatted reports.

A data model for OLAP applications should have certain characteristics in order to support the complex data found in many real-world systems. We present nine advanced requirements that a multidimensional data model should satisfy and illustrate the requirements using a real-world case study from the clinical world. We present an extended multidimensional data model that addresses all nine requirements. The data model supports modeling explicit hierarchies in the dimensions, to aid the user in navigating the data. Multiple hierarchies in each dimension is supported, to allow for different aggregation paths, and the non-strict hierarchies found in real-world dimensions, i.e., where a dimension item may have several parents, are also supported. The model treats dimensions and measures symmetrically, to allow measures to be used as dimensions and vice versa. Many-to-many relationships between facts and dimensions can be captured directly in the model, which is important as these relationships often occur in real-world data, e.g., a patient may have several diagnoses. The data model supports getting correct results when aggregating data, e.g., data will not be double-counted and non-additive data cannot be added. Data change over time, so support for handling change and time is part of the model. Aspects of the uncertainty often associated with data are also handled by the model. Finally, the model supports handling data with different levels of granularity, which is a need in some applications.

The model is equipped with an algebra that is closed and at least as strong as relational algebra with aggregation functions. Finally, we outline how the model can be implemented using relational databases.

Eight previously proposed data models, which are representative for the spectrum of multidimensional data models, are evaluated against the nine requirements, and it is shown that no other model satisfies more than four of these requirements. Importantly, no other model supports many-to-many relationships between facts and dimensions, handling of uncertainty, and different levels of granularity at all, and no other model completely supports handling change and time or non-strict hierarchies.

The presentation is structured as follows. Section 2 sets the stage by first presenting a real-world case study from the clinical world together with nine requirements to multidimensional data models; it then describes and evaluates previously proposed models against the requirements. Section 3 proceeds to first define the basic extended multidimensional data model, using examples from the case study for illustration, then adds support for handling time and uncertainty to the model. With the data structures of the model available, Section 4 defines an algebra for the model and discusses its properties. Section 5 evaluates the model against the requirements, and Section 6 summarizes and points to future directions. An appendix outlines how to implement the model using relational databases.

2 Motivation

This section illustrates the shortcomings of the previously proposed multidimensional models. First, we present a case study that shows some of these limitations. The case is taken from the domain of healthcare, where we look at patients, their diagnoses, and their place of residence. Second, we list the requirements for features that a data model should satisfy in order to meet the needs of the case study. Third, we relate these requirements to the existing multidimensional data models.

2.1 A Case Study

The case study concerns the patients in a hospital, their associated diagnoses, and their place of residence. The goal is to investigate whether some diagnoses occur more often in some areas than in others, in which case environmental or lifestyle factors might be contributing to the disease pattern. An ER diagram illustrating the underlying data is seen in Figure 1.

The most important entities are the *patients*. For a patient, we record Name, Social Security Number (SSN), and Date of Birth. From the Date of Birth and the current date, we can derive the Age attribute, which is parenthesized to show that it is derived.

Each patient can have one or more *diagnoses*. The attribution of diagnoses to patients can vary over time, and we also record the time interval where a diagnosis is considered to be valid for a patient. We also record the *type of diagnostization*, to show whether a diagnosis is considered to be *primary* or *secondary*. A primary diagnosis is considered to be the most important reason for a treatment, while secondary diagnoses complete the view of the patient's condition. A patient may have only one primary diagnosis at any one point in time.

When registering a diagnosis of a patient, physicians often use different levels of granularity. Some will use the very precise diagnosis "Insulin dependent diabetes," while others will use the more imprecise diagnosis "Diabetes," which covers a wider range of patient conditions, corresponding to a number of more precise diagnoses. To model this, the relationship from patient to diagnoses is to the supertype "Diagnosis." The Diagnosis type has three subtypes, corresponding to different levels of granularity, the *low-level diagnosis*, the *diagnosis family*, and the *diagnosis group*. Examples of these are "Insulin dependent diabetes during pregnancy," "Insulin dependent diabetes," and "Diabetes," respectively. The higher-level diagnoses are both (imprecise) diagnoses in their own right, but also function as groups of lower-level diagnoses, as

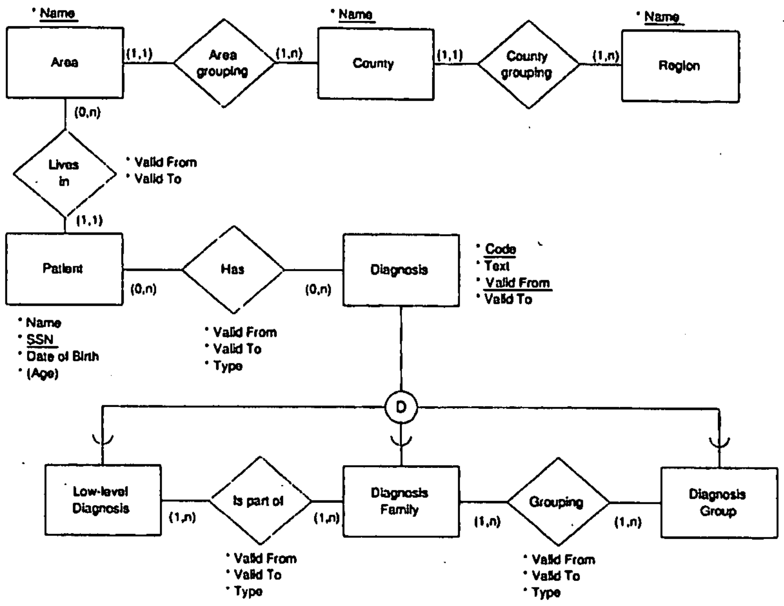


Figure 1: Patient Diagnosis Case Study

will be discussed later.

For diagnoses, we record an alphanumeric code and a descriptive text. The code and text are usually determined by a standard classification of diseases, e.g., the World Health Organization's International Classification of Diseases (ICD-10) [13], but we also allow user-defined diagnoses.

Over time, medical knowledge evolves, and a disease classification reflects this by changing its contents. What often happens is that a diagnosis is superseded by one or more new diagnoses that better reflect the current understanding of this particular medical condition. To model this fact, we associate with each diagnosis a *period of validity*, represented by the attributes *Valid From* and *Valid To*. This period of validity is the time interval in the real world where the diagnosis can be used for diagnostization, and has the associated code and text. The classifications evolve only slowly, so the granularity of time used can be quite high, e.g., days.

As there are several thousand diagnoses in the classification, the diagnoses are grouped into *diagnosis families* and these in turn into *diagnosis groups*, thus creating a hierarchy in the classification. We have two types of hierarchies: the standard hierarchy determined by the classification owner, e.g., the WHO, and the user-defined hierarchy, which is used by physicians to group diagnoses on an ad-hoc basis in other ways than the standard classification allows.

First, the hierarchy groups low-level diagnoses into *diagnosis families*, each of which consists of 5–50 related diagnoses. For example, the diagnosis "Insulin dependent diabetes during pregnancy[1]" is part of the family "Diabetes during pregnancy." In the standard classification, a low-level diagnosis is part of exactly one diagnosis family. However, physicians often have a need to group diagnoses in other ways than the standard allows, so we also allow a *user-defined* hierarchy. The *Type* attribute on the relationship

[1]The reason for having a separate pregnancy related diagnosis is that diabetes must be monitored and controlled particularly intensely during a pregnancy to assure good health of both mother and child.

determines whether the relation between two entities is part of the standard or the user-defined hierarchy.

Thus, a diagnosis can be part of several diagnosis families, e.g., the "Insulin dependent diabetes during pregnancy" diagnosis is part of both the "Diabetes during pregnancy" and the "Insulin dependent diabetes" family. The participation of individual diagnoses in a family may change over time, so we record the time interval during which a diagnosis is part of a family.

ID	Name	SSN	Date of Birth
1	John Doe	12345678	25/05/69
2	Jane Doe	87654321	20/03/50

Patient Table

PatientID	DiagnosisID	ValidFrom	ValidTo	Type
1	9	01/01/89	NOW	Primary
2	3	23/03/75	24/12/75	Secondary
2	8	01/01/70	31/12/81	Primary
2	5	01/01/82	30/09/82	Secondary
2	9	01/01/82	NOW	Primary

Has Table

ID	Code	Text	ValidFrom	ValidTo
3	P11	Diabetes during pregnancy	01/01/70	31/12/79
4	O24	Diabetes during pregnancy	01/01/80	NOW
5	O24.0	Insulin dependent diabetes during pregnancy	01/01/80	NOW
6	O24.1	Non insulin dependent diabetes during pregnancy	01/01/80	NOW
7	P1	Other pregnancy related diseases	01/01/70	31/12/79
8	D1	Diabetes	01/10/70	31/12/79
9	E10	Insulin dependent diabetes	01/01/80	NOW
10	E11	Non insulin dependent diabetes	01/01/80	NOW
11	E1	Diabetes	01/01/80	NOW
12	O2	Other pregnancy related diseases	01/10/80	NOW

Diagnosis Table

ParentID	ChildID	ValidFrom	ValidTo	Type
4	5	01/01/80	NOW	WHO
4	6	01/01/80	NOW	WHO
7	3	01/01/70	31/12/79	WHO
8	3	01/01/70	31/12/79	User-defined
9	5	01/01/80	NOW	User-defined
10	6	01/01/80	NOW	User-defined
11	9	01/01/80	NOW	WHO
11	10	01/01/80	NOW	WHO
12	4	01/01/80	NOW	WHO

Grouping Table

Table 1: Data for the Case Study

Second, the diagnosis families are grouped into *diagnosis groups*, consisting of 5–25 families, and one family may be part of several groups. For example, the family "Diabetes during pregnancy" may the part of the "Diabetes" and the "Other pregnancy related diseases" groups. In the standard hierarchy, however, a family belongs to exactly one group. Here we also use the *Type* attribute to distinguish between the standard and the user-defined hierarchy. The grouping of families into groups can also change over time, so we record the time interval during which a family is part of a group.

In the standard hierarchy, a lower-level item belongs to exactly one higher-level item, thus the standard hierarchy is a *strict, partitioning* hierarchy. In the user-defined hierarchy, a lower-level item can be a member of zero or more higher-level items, making it a *non-strict, non-partitioning* hierarchy. Properties of the hierarchies will be discussed in more detail in Section 3.4.

We also record the place of residence for the patients. A patient may only live in one place at any one point in time. When people move, their previous address is still interesting, so we also record the associated period of residence. We record the place of residence at the granularity of an *area*, which designates a small, bounded area of a few square kilometers. An area is part of exactly one *county*, which in turn is part of exactly one *region*. Thus, we have a *strict, partitioning* hierarchy. For areas, counties, and regions we just record the name.

In order to list some example data, we assume a standard mapping of the ER diagram to relational tables, i.e., one table per entity type, one-to-many relationships handled using foreign keys, and many-to-many relationships handled using separate tables. Relationships that change over time are also handled using separate tables. We also assume the use of surrogate keys, named *ID*, with globally unique values. Dates are written in the format dd/mm/yy. For the *Valid To* attribute, we use the special value "NOW" value that denotes the current time[2] [25]. As the three subtypes of the Diagnosis type do not have any attributes of their own, all three are mapped to a common Diagnosis table. The "is part of" and "grouping" relationships are also mapped to a common "Grouping" table. The data consists of two patients, four diagnostizations, and 10 diagnoses in a hierarchy. On January 1, 1980, a new, more detailed classification with a new coding scheme is introduced. The resulting tables are shown in Table 1 and will be used in examples throughout the paper.

2.2 Requirements for Data Analysis

This section describes the features that a data model should possess in order to fully support our sample case and other advanced uses. Current multidimensional models are evaluated against these features in the next section.

1. *Explicit hierarchies in dimensions.* The hierarchies in the dimensions should be captured explicitly by the schema, so the user has available the relation between the different levels in the hierarchy. In our example, the hierarchies $diagnosis < family < group$ and $area < county < region$ should be captured.

2. *Symmetric treatment of dimensions and measures.* The data model should allow measures to be treated as dimensions and vice versa. In our example, the attribute Age for patients would typically be treated as a measure, to allow for computations such as average age, etc., but we should also be able to define an Age dimension which allows us to group the patients into age groups.

3. *Multiple hierarchies in each dimension.* In one dimension, there can be more than one path along which to aggregate data. As an example, let us assume that we have a Time dimension on the Date of Birth attribute. Days roll up to weeks and to months, but weeks do not roll up to months. To model this, multiple hierarchies in each dimension are needed.

4. *Support for correct aggregation.* The data model should support getting results that are "correct," i.e., meaningful to the user, when aggregating data. One aspect of this is to avoid double-counting of data. In our case study, when asking for the numbers of patients in different diagnosis groups, we should only count the same patient once per group, even though the patient has several diagnoses in a group. The user should also be able to specify what aggregations are considered meaningful for the different kinds of data available, and the model should provide a foundation for enforcing these specifications. For example, it may not be meaningful to add inventory levels together, but performing average

[2]Note that this value is *dynamic*, i.e., it continues to grow.

calculations on them does make sense. In the field of statistical databases, a closely related concept is *summarizability* [7, 8], which means that an aggregate result, e.g., total sales, can be computed by directly combining results from lower-level aggregations, e.g., the sales for each store.

5. *Non-strict hierarchies.* The hierarchies in a dimension are not always strict, i.e., we can have many-to-many relationships between the different levels in a dimension. In our example, the diagnosis hierarchy is not strict. The data model should be able to handle these just as well as "ordinary" strict dimensions.

6. *Many-to-many relationships between facts and dimensions.* The relationship between fact and dimension is not always the classical many-to-one mapping. In our case study, the same patient may have several diagnoses, even at the same point in time.

7. *Handling change and time.* Data change over time, but we should be able to get meaningful analysis results across changes. In the example, one diagnosis can be superseded by two new ones, but patients are still diagnostizised with the old one. It should be possible to easily combine data across changes. The problem typically referred to as handling *slowly changing dimensions* [4, 20] is part of this problem.

8. *Handling uncertainty.* In our case study, one diagnosis is superseded by two new ones. We know that in 90% of the cases where we used the old diagnosis, we will now use the first of the new diagnoses. Thus, when requesting data grouped by diagnosis for a period that spans the change, we want the old diagnosis to be counted together with the first new diagnosis. The data model should allow expressing this and also support giving some kind of indications in the query results, indicating how many of these "converted" diagnoses are counted in the result.

9. *Handling different levels of granularity.* Fact data might be registered at different granularities. In our example, the diagnosis of a diabetes patient may be registered differently by different physicians. Some will use a very specific diagnosis such as "Insulin dependent diabetes," while others will use the more imprecise "Diabetes," which covers several lower-level diagnoses. It should still be possible to get correct analysis results when data is registered at different granularities.

2.3 Related Work

In this section we evaluate data models that have previously been proposed for data warehousing according to the requirements in the previous section.

We consider the models of Rafanelli & Shoshani [7], Agrawal et al. [6], Gray et al. [3], Kimball [4], Li & Wang [11], Gyssens & Lakshmanan [10], Datta & Thomas [14], and Lehner [12]. The models can be divided into three groups: *simple cube models, structured cube models,* and *statistical objects.*

The simple cube models [3, 4, 10, 14] treat data as n-dimensional cubes. Generally, the data is divided into *facts*, or *measures*, e.g., Age, on which calculations should be performed, and *dimensions*, e.g., Diagnosis, which characterize the facts. Each dimension has a number of attributes, which can be used for selection and grouping. In our example, a "Residence" dimension having the attributes "Area," "County," and "Region" would be used to characterize the patients. The hierarchy between the attributes is not captured explicitly by the schema of the simple cubes, so the user will not be able to learn from the schema that Area rolls up to County and not the other way around. Star schema designs [4] are also considered as a simple cubes, as they are semantically equivalent to these.

The structured cube models [6, 11, 12] capture the hierarchies in the dimensions explicitly, providing better guidance for the user navigating the cubes. This information may also be useful for query optimization

[15]. The hierarchies are captured using either *grouping relations* [11], *dimension merging functions* [6], or an explicit tree-structured hierarchy as part of the cube [12].

The last group of models is the *statistical objects* [7]. For this group, a structured classification hierarchy is coupled with an explicit aggregation function on a single measure to produce a "pre-cooked" object that will answer a very specific set of questions. This approach is not as flexible as the others, but unlike most of these, it provides some protection (summarizability) against getting incorrect results from queries.

The results of evaluating the eight data models against our nine requirements are seen in Table 2. If a model supports all aspects of a requirement, we say that the model provides *full* support, denoted by "√". If a model supports some, but not all, aspects of a requirement, we say that it provides *partial* support, denoted by "p". When it is not possible for the authors to determine how support for a requirement should be accomplished in the model, we say that the model provides *no* support, denoted by "-".

	1	2	3	4	5	6	7	8	9
Rafanelli & Shoshani [7]	√	-	-	√	p	-	-	-	-
Agrawal et al. [6]	p	√	√	-	p	-	-	-	-
Gray et al. [3]	-	√	√	p	-	-	-	-	-
Kimball [4]	-	-	√	p	-	-	p	-	-
Li & Wang [11]	p	-	√	p	-	-	-	-	-
Gyssens & Lakshmanan [10]	-	√	√	p	-	-	-	-	-
Datta & Thomas [14]	-	√	√	-	p	-	-	-	-
Lehner [12]	√	-	-	√	-	-	-	-	-

Table 2: Evaluation of the Data Models

1. *Explicit hierarchies in dimensions*: The simple cube models [3, 4, 10, 14] do not capture the hierarchies in the dimensions explicitly. Some models provide partial support by the *grouping relation* [11] and *dimension merging function* [6] constructs, but do not capture the complete hierarchy together with the cube. This is done by the last two models [7, 12], thus capturing the full cube navigation semantics in the schema.

2. *Symmetric treatment of dimensions and measures*: Half of the models [4, 7, 11, 12] distinguish sharply between measures and dimensions. An attribute designated as a measure cannot be used as a dimensional attribute and vice versa. This restricts the flexibility of the cube designs, e.g., if the Age attribute of the example is a measure, it cannot be used to group patients into age groups. The other half of the models [3, 6, 10, 14] do not impose this restriction. They either do not distinguish between measures and dimensions [3, 10], or they allow for the conversion of measures to dimensions and vice versa [6, 14].

3. *Multiple hierarchies in each dimension*: Some models [7, 12] require that the dimension hierarchies are tree-structured. To support multiple hierarchies, a more general lattice structure is required. All the other models [3, 4, 6, 10, 11, 14] allow multiple hierarchies.

4. *Support for correct aggregation*: Half of the models [3, 4, 10, 11] support correct aggregation partially, by implicitly requiring the dimension hierarchies to be *strict* and *partitioning*, i.e., a lower-level item maps to exactly one item on the next level. This is one of the conditions of summarizability [8]. Two of the models allow for non-strict hierarchies, while not addressing the issue of double-counting, thus providing no support [6, 14]. The remaining two models [7, 12] place explicit conditions on both

the hierarchy (strict and partitioning) and the aggregation functions used (only additive data may be added, etc.), thus providing full support for correct aggregation.

5. *Non-strict hierarchies*: Most of the models [3, 4, 10, 11, 12] implicitly or explicitly require that hierarchies be strict. Two models [6, 14] mention briefly that non-strict hierarchies are allowed, but does not go deeper into the issues raised by allowing this, e.g., the possibility of double-counting and the use of pre-computed aggregates. The remaining model [7] investigates the possible problems with allowing non-strict hierarchies and advises against using this feature.

6. *Many-to-many relationships between facts and dimensions*: None of the models allow many-to-many relationships between facts and their associated dimensions, such as the relationship between patients and diagnoses in the example.

7. *Handling change and time*: Only one model [4] discusses this issue, but none the proposed solutions fully support analysis across changes in the dimensions. None of the other models support analysis across changes, although one mention that this is a very important issue [12].

8. *Handling uncertainty*: None of the models provide built-in support for uncertainty in the data.

9. *Handling different levels of granularity*: None of the models handle different levels of granularity in the data.

To conclude, the models generally provide full or partial support for most of requirements 1–4. Requirement 5 (non-strict hierarchies) is partially supported by three of the models, while requirement 7 (handling change and time) is only partially supported by Kimball [4]. Requirements 6, 8, and 9 are not supported by any of the models. The objective of the model proposed in this paper is to support all nine requirements.

3 An Extended Multidimensional Data Model

In this section we define our model. For every part of the model, we define the *intension*, the *extension*, and give an illustrating example. To avoid unnecessary complexity, we first define the basic model and then define extensions for handling time and uncertainty later.

3.1 The Basic Model

An *n-dimensional fact schema* is a two-tuple $\mathcal{S} = (\mathcal{F}, \mathcal{D})$, where $\mathcal{F}$ is a *fact type* and $\mathcal{D} = \{\mathcal{T}_i, i = 1, .., n\}$ is its corresponding *dimension types*.

Example 1 In the case study from Section 2.1 we will have *Patient* as the fact type, and *Diagnosis, Residence, Age, Date of Birth (DOB), Name*, and *Social Security Number (SSN)* as the dimension types. The intuition is that *everything* that characterizes the fact type is considered to be *dimensional*, even attributes that would be considered as *measures* in other models.

A dimension type $\mathcal{T}$ is a four-tuple $(\mathcal{C}, \leq_{\mathcal{T}}, \top_{\mathcal{T}}, \bot_{\mathcal{T}})$, where $\mathcal{C} = \{\mathcal{C}_j, j = 1, .., k\}$ are the *category types* of $\mathcal{T}$, $\leq_{\mathcal{T}}$ is a partial order on the $\mathcal{C}_j$'s, with $\top_{\mathcal{T}} \in \mathcal{C}$ and $\bot_{\mathcal{T}} \in \mathcal{C}$ being the top and bottom element of the ordering, respectively. Thus, the category types form a lattice. The intuition is that one category type is "greater than" another category type if members of the former's extension logically contain members of the latter's extension, i.e., they have a larger element size. The top element of the ordering corresponds to the largest possible element size, that is, there is only one element in it's extension, logically containing all other elements.

We say that C_j *is a category type of* $\mathcal{T}$, written $C_j \in \mathcal{T}$, if $C_j \in \mathcal{C}$. We assume a function $Pred : \mathcal{C} \mapsto 2^{\mathcal{C}}$ that gives the set of immediate predecessors of a category type C_j.

Example 2 Low-level diagnoses are contained in diagnosis families, which are contained in diagnosis groups. Thus, the *Diagnosis* dimension type has the following order on its category types: $\perp_{Diagnosis}$ = *Low-level Diagnosis* < *Diagnosis Familiy* < *Diagnosis Group* < $\top_{Diagnosis}$. We have that $Pred$(*Low-level Diagnosis*) = {*Diagnosis Family*}. Other examples of category types are *Age* and *Ten-year Age Group* from the Age dimension type, and *DOB* and *Year* from the DOB dimension type. Figure 2, to be discussed in detail later, illustrates the dimension types of the case study.

Many types of data, e.g., ages or sales amounts, can be added together to produce meaningful results. This data has an ordering on it, so computing the average, minimum, and maximum values make sense. For other types of data, e.g., dates of birth or inventory levels, the user may not find it meaningful in the given context to add them together. However, the data has an ordering on it, so taking the average, or computing the maximum or minimum values do make sense. Some types of data, e.g., diagnoses, do not have an ordering on them, and so it does not make sense to compute the average, etc. Instead, the only meaningful aggregation is to count the number of occurrences.

We can support correct aggregation of data by keeping track of what types of aggregate functions can be applied to what data. This information can then be used to either prevent users from doing "illegal" calculations on the data completely, or to warn the users that the result might be "wrong," e.g., the same patient is counted twice, etc. In line with this reasoning and previous work [12, 19], we distinguish between three types of aggregate functions: Σ, applicable to data that can be added together, ϕ, applicable to data that can be used for average calculations, and c, applicable to data that is constant, i.e., it can only be counted. Considering only the standard SQL aggregation functions, we have that Σ = {SUM, COUNT, AVG, MIN, MAX}, ϕ = {COUNT, AVG, MIN, MAX}, and c = {COUNT}. The aggregation types are ordered, $c \subset \phi \subset \Sigma$, so data with a higher aggregation type, e.g., Σ, also possess the characteristics of the lower aggregation types. For each dimension type $\mathcal{T} = (\mathcal{C}, \leq_{\mathcal{T}})$, we assume a function $Aggtype_{\mathcal{T}} : \mathcal{C} \mapsto \{\Sigma, \phi, c\}$ that gives the aggregation type for each category type.

Example 3 In the case study, *Aggtype*(*Low-level Diagnosis*) = c, *Aggtype*(*Age*) = Σ, *Aggtype*(*Ten-year Age Group*) = c, and *Aggtype*(*DOB*) = ϕ.

A *dimension* D of type $\mathcal{T} = (\{C_j\}, \leq_{\mathcal{T}}, \top_{\mathcal{T}}, \perp_{\mathcal{T}})$ is a two-tuple $D = (C, \leq)$, where $C = \{C_j\}$ is a set of *categories* C_j such that $Type(C_j) = C_j$ and $\leq$ is a partial order on $\cup_j C_j$, the union of all dimension values in the individual categories. A category C_j of type C_j is a set of *dimension values* e such that $Type(e) = C_j$.

The definition of the partial order is: given two values e_1, e_2 then $e_1 \leq e_2$ if e_1 is logically contained in e_2. We say that C_j is a category of D, written $C_j \in D$, if $C_j \in C$. For a dimension value e, we say that e is a dimensional value of D, written $e \in D$, if $e \in \cup_j C_j$.

The category type $\perp_{\mathcal{T}}$ in dimension type $\mathcal{T}$ contains the values with the smallest value size. The category type with the largest value size, $\top_{\mathcal{T}}$, contains exactly one value, denoted $\top$. For all values e of the category types of D, $e \leq \top$. Value $\top$ is similar to the *ALL* construct of Gray et al. [3].

Example 4 In our *Diagnosis* dimension we have the following categories, named by their type. *Low-level Diagnosis* = $\{3, 5, 6\}$, *Diagnosis Family* = $\{4, 7, 8, 9, 10\}$, *Diagnosis Group* = $\{11, 12\}$, and $\top_{Diagnosis}$ = $\{\top\}$. The values in the sets refer to the *ID* field in the Diagnosis table of Table 1. The partial order $\leq$ is given by the first two columns in the Grouping table in Table 1. Additionally, the top value $\top$ is greater than, i.e., logically contains, all the other diagnosis values.

We say that the dimension $D' = (C', \leq')$ is a *subdimension* of the dimension $D = (C, \leq)$ if $C' \subseteq C$ and $e_1 \leq' e_2 \Leftrightarrow \exists C_1, C_2 \in C'(e_1 \in C_1, e_2 \in C_2 \wedge e_1 \leq e_2)$, that is, D' has a subset of the categories of D and $\leq'$ is the restriction of $\leq$ to these categories. We note that D is a subdimension of itself.

Example 5 We obtain a subdimension of the Diagnosis dimension from the previous example by removing the *Low-level Diagnosis* and *Diagnosis Family* categories, retaining only *Diagnosis Group* and $\top_{Diagnosis}$.

It is desirable to distinguish between the dimension values in themselves and the real-world "names" that we use for them. The names might change or the same value might have more than one name, making the name a bad choice for identifying an value. In common database terms, this is the argument for *object ids* or *surrogates*.

To support this feature, we require that a category C has one or more *representations*. A representation Rep is a bijective function $Rep : Dom(C) \leftrightarrow Dom_{Rep}$, i.e., a value of a representation uniquely identifies a single value of a category and vice versa, thus making the representation an "alternate key." We use the notation $Rep(e) = v$ to denote the mapping from dimension values to representation values.

Example 6 A diagnosis value has two representations, *Code* and *Text*. Using the ID's from the Diagnosis table to identify the values, we have $Code(3) =$ "O24" and $Text(3) =$ "Diabetes during pregnancy."

Let F be a set of facts, and $D = (\{C_j\}, \leq)$ a dimension. A *fact-dimension relation* between F and D is a set $R = \{(f, e)\}$, where $f \in F$ and $e \in \cup_j C_j$. Thus R links facts to dimension values. We say that fact f is *characterized by* dimension value e, written $f \rightsquigarrow e$, if $\exists e_1 \in D\ ((f, e_1) \in R \wedge e_1 \leq e)$. We require that $\forall f \in F\ (\exists e \in \cup_j C_j\ ((f, e) \in R))$; thus we do not allow missing values. The reasons for disallowing missing values are that they complicate the model and often have an unclear meaning. If it is unknown which dimension value a fact f is characterized by, we add the pair $(f, \top)$ to R, thus indicating that we cannot characterize f within the particular dimension.

Example 7 The fact-dimension relation R links patient facts to diagnosis dimension values as given by the Has table from the case study. Leaving out the temporal aspects for now, we get that $R = \{(1,9), (2,3), (2,5), (2,8), (2,9)\}$. Note that we can relate facts to values in higher-level categories, e.g., fact 1 is related to diagnosis 9, which belongs to the *Diagnosis Family* category. Thus, we do not require that e belongs to $\perp_{Diagnosis}$, as do the existing data models. If no diagnosis is known for patient 1, we would have added the pair $(1, \top)$ to R.

A *multidimensional object* (MO) is a four-tuple $M = (\mathcal{S}, F, D, R)$, where $\mathcal{S} = (\mathcal{F}, \mathcal{D} = \{\mathcal{T}_i\})$ is the fact schema, $F = \{f\}$ is a set of *facts* f where $Type(f) = \mathcal{F}$, $D = \{D_i, i = 1, .., n\}$ is a set of *dimensions* where $Type(D_i) = \mathcal{T}_i$, and $R = \{R_i, i = 1, .., n\}$ is a set of fact-dimension relations, such that $\forall i((f, e) \in R_i \Rightarrow f \in F \wedge \exists C_j \in D_i(e \in C_j))$.

Example 8 For the case study, we get a six-dimensional MO $M = (\mathcal{S}, F, D, R)$, where $\mathcal{S} =$ (*Patient*, {*Diagnosis, DOB, Residence, Name, SSN, Age*}) and $F = \{1, 2\}$. The definition of the diagnosis dimension and its corresponding fact-dimension relation was given in the previous examples. Due to space constraints, we do not list the contents of the other dimensions and fact-dimension relations, but just outline their structure. The Name and SSN dimensions are simple, i.e., they just have a $\perp$ category type, Name respectively SSN, and a $\top$ category type. The Age dimension groups ages (in years) into five-year and ten-year groups, e.g., 10–14 and 10–19. The Date-of-Birth dimension has two hierarchies in it: days are grouped into weeks, or days are grouped into months, with the further levels of quarters, years, and decades. We will refer to this MO as the "Patient" MO. A graphical illustration of the schema of the "Patient" MO is seen in Figure 2.

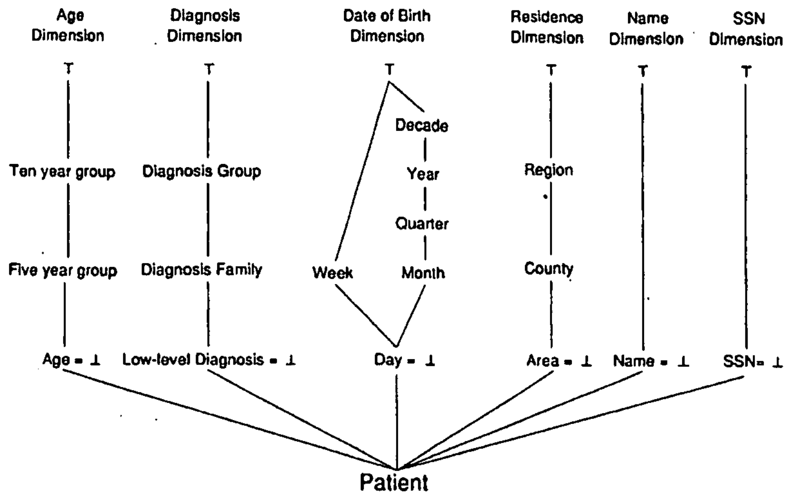


Figure 2: Schema of the Case Study.

A collection of multidimensional objects, possibly with shared subdimensions, is called a *multidimensional object family.*

Example 9 To illustrate the usefulness of shared subdimensions in multidimensional object families, imagine performing the following steps. Create a subdimension of the Diagnosis dimension that includes only *Diagnosis Group* and $\top_{Diagnosis}$, and a subdimension of the Age dimension that includes only *Ten-Year Group* and $\top_{Age}$. Make an MO with these two dimensions and the fact type Patient for all patients in the country. This results in an MO capturing all patients in the country together with their diagnosis groups and their ten-year age groups. Putting this MO together with the "Patient" MO from the example above, we obtain a multidimensional object family with two shared subdimensions. The shared subdimensions could be used to investigate how diagnoses versus age groups for the patient group from the case study compare to the national average.

To summarize the essence of our model, the facts are objects with a *separate identity*. Thus, we can test facts for equality, but we do not assume an ordering on the facts. The combination of dimensions values that characterize the facts of a fact set is *not* a "key" for the fact set. Thus, we may have "duplicate values," in the sense that several facts may be characterized by the same combination of dimension values. But, the facts of an MO is a *set*, so we do not have duplicate *facts* in an MO.

3.2 Handling Time

We now investigate how to build temporal support into the model. The vast majority of research in temporal data models assumes a discrete time domain (for example, most data models in the most recent collection of temporal database papers [16] explicitly assume a discrete model of time). Also the temporal data types offered by the SQL standard [17] are discrete and bounded. Thus, we assume a time domain that is discrete and bounded, i.e., isomorphic with a bounded subset of the natural numbers. The values of the time domain are called *chronons*. They correspond to the finest granularity in the time domain [22]. We let T, possibly subscripted, denote a set of chronons.

The *valid time* of a statement is the time when the statement is true in the modeled reality [1]. Valid time is very important to capture because the real world is where the users reside, and we *allow* the attachment of valid time to the data, but do not require it. If valid time is not attached to the data, we assume the data to be *always* valid. If valid time is attached to an MO, we call it a *valid-time* MO.

In general, valid time may be assigned to anything that has a truth value. In our model, this is the partial order between dimension values, the mapping between values and representations, the fact-dimension relations, and the membership of values in categories. It is important to be able to capture all these aspects.

We add valid time to the dimension partial order $\leq$ by adding T_v, the set of chronons during which the relation holds in the real world, to each relation between two values. We write that $e_1 \leq_{T_v} e_2$ if $e_1 \leq e_2$ during each chronon in T_v. The partial order $\leq_{T_v}$ has the following property: $e_1 \leq_{T_{1_v}} e_2 \wedge e_2 \leq_{T_{2_v}} e_3 \Rightarrow e_1 \leq_{T_{1_v} \cap T_{2_v}} e_3$. Similarly, we write $Rep(e) =_{T_v} v$ to denote that the representation Rep of the value e has value v during each chronon in T_v. For each fact-dimension relation between a fact f and a dimension value e, we capture the set of chronons T_v when the two are related. We write $(f,e) \in_{T_v} R$ when $(f,e) \in R$ during each chronon in T_v. We use the notation $f \leadsto_{T_v} e$ when $(f,e') \in_{T_v} R \wedge e' \leq_{T_v} e$. Finally, we add valid time to membership of dimension values in categories, writing $e \in_{T_v} C$ when $e \in C$ during each chronon in T_v.

The set of chronons that is attached to a statement is the *maximal* set of chronons when the statement is valid, so the data is always "coalesced" [1]. Thus, we do not have the problem of "value-equivalent" statements [1, 21, 23], where the same statement appears several times with different times attached to it, e.g., $e_1 \leq_{T_1} e_2$ and $e_1 \leq_{T_2} e_2$, where $T_1 \neq T_2$. However, by implication, statements are valid for any subset of their attached time, e.g., $T_1 \subseteq T_2 \wedge e_1 \leq_{T_2} e_2 \Rightarrow e_1 \leq_{T_1} e_2$.

Example 10 For our examples, we use interval notation for T_v, with the chronon size equal to Day. For the partial order for the Diagnosis dimension, we have $7 \leq_{[01/01/70-31/12/79]} 3$. For the representation, we have $Code(8)_{[01/01/70-31/12/79]} = \text{D1}$. For the fact-dimension relation, we have $(2,3) \in_{[23/03/75-24/12/75]} R$. For the category membership, we have $10 \in_{[01/01/80-NOW]}$ *Diagnosis Family*.

To sum up, by extending the dimension partial order with links between dimension values that represent the "same" thing across change, we have a foundation for handling analysis across changes. This allows us to obtain meaningful results when we analyze data across changes in the dimension.

Example 11 When looking at the data from the current point in time, we want to count the patients diagnosed with the old "Diabetes" diagnosis ($ID = 8$) together with those diagnoses with the new "Diabetes" diagnosis ($ID = 11$) when we look at diagnostizations from 1970 to the present. This is done by defining that $8 \leq_{[01/01/80-NOW]} 11$, i.e., from 1980 up till now, we consider the diagnosis 8 to be logically contained in the diagnosis 11.

Valid time is not the only temporal aspects that may be interesting to our model. It is also interesting to capture when statements are present in the database, as the time a statement is present in the database almost never corresponds to the time it is true in the real world. We need to know when data are present in the database for accountability and traceability purposes.

The *transaction time* of a statement is the time when the statement is current in the database and may be retrieved [1]. Generally, transaction time can be attached to anything that valid time can be attached to. The addition of transaction time is orthogonal to the addition of valid time. Additionally, transaction time can be added to data that does not have a truth value. In our model, we could record when facts, e.g., patients, are present in the database. We do not think that this is very interesting in itself, as facts are only interesting when they participate in fact-dimension relations. Thus, we do not record this.

If transaction time is attached to an MO, we call it a *transaction-time* MO. If both valid and transaction time is attached to an MO, we call it a *bitemporal* MO. If no time is attached to an MO, we call it a *snapshot*

MO. In our notation, we use T_t to denote the set of chronons when data is current in the database. We use $T_t \times T_v$ to denote sets of bitemporal chronons.

3.3 Handling Uncertainty

Sometimes the data available contains uncertainties that cannot adequately be captured using standard techniques such as default values, etc. Instead, we handle this uncertainty explicitly in our data model. To do so, we introduce measures of probability. In general, it makes sense to attach a probability to a statement if the statement can be given a valid time. In our model, this applies to dimension partial orders, fact-dimension relations, mappings between values and representations, and category memberships for values. However, we attach probabilities to the former two only. It is of little use to have a probability for the mapping between values and representations, which would violate the requirement that a representation is an alternate key. Also, giving probabilities to the membership of categories is omitted, as values belong fully to one category at any given time.

First, we add probability to the partial order on dimension values. Given two dimension values e_1, e_2 and a number p such that $0 \leq p \leq 1$, we write that $e_1 \leq_p e_2$ if $e_1 \leq e_2$ with probability p. Second, we add probability to a fact-dimension relation R. Given a fact f, a dimension value e and a number p such that $0 \leq p \leq 1$, we write that $(f, e) \in_p R$ if $(f, e) \in R$ with probability p. We write $f \leadsto_p e$ if $(f, e') \in_{p_1} R \wedge e' \leq_{p_2} e \wedge p = p_1 \cdot p_2$. Note that a probability is assigned to *all* (ancestor,descendent) links in the partial order, not just the direct (parent,child) links.

Example 12 Before 1980, diagnostizations in the case study do not differentiate between insulin dependent and non insulin dependent diabetes. From 1980 and on this is the case. Suppose that we know that 90% of the diagnostizations made with the old "Diabetes" diagnosis were for insulin dependent diabetes cases. When looking for the number of insulin dependent diabetes patients from 1970 up till now, we want to count the old "Diabetes" diagnostizations too. We do this by extending the Diagnosis partial order with the information that $8 \leq_{0.9} 9$.

Suppose that physicians are allowed to express their belief in the correctness of a diagnostization by attaching a probability p to it. In the case study, the physician is 95% certain that John Doe ($ID = 1$) has insulin dependent diabetes ($ID = 9$). Thus, for the fact-dimension relation R, $(1, 9) \in_{0.95} R$.

To summarize, the addition of uncertainty to the model is orthogonal to the features for handling time, thus any combination of extensions is valid. If both time and probabilities are added to an MO, we assign a probability p_t for *each* chronon t in the chronon set T. This is done to avoid the problems of value-equivalent tuples. However, interval notation such as $8 \leq_{0.9[1980-NOW]} 9$ is used in examples. If probability is assigned to an MO, we call it a *probabilistic* MO. If no probability is assigned, we call it a *deterministic* MO.

3.4 Properties of the Model

In this section important properties of the model that relate to the use of pre-computed aggregates is defined and discussed. The first important concept is *summarizability*, which intuitively means that individual aggregate results can be combined directly to produce new aggregate results.

Definition 1 Given a type T, a set $S = \{S_j, j = 1, .., k\}$, where $S_j \in 2^T$, and a function $g : 2^T \mapsto T$, we say that g is *summarizable* for S if $g(\{\{g(S_1), .., g(S_k)\}\}) = g(S_1 \cup .. \cup S_k)$. The set of arguments on the left side of the equation is a multi-set, or bag, i.e., the same result value can occur multiple times.

Summarizability is an important concept as it is a condition for the flexible use of pre-computed aggregates. Without summarizability, lower-level results generally cannot be directly combined into higher-level results. This means that we cannot choose to pre-compute only a relevant selection of the possible aggregates and then use these to compute higher-level aggregates on-the-fly. Instead, we have to pre-compute the total results for all the aggregations that we need fast answers to, while other aggregates must be computed from the base data. Space and time constraints can be prohibitive for pre-computing all results, while computing aggregates from scratch results in long response times. In this case, an attractive alternative is the use of *sampling* techniques to answer the queries [24]. Using sampling, only a small sample of the available data is read and used to *estimate* the result of the query. This can produce very fast response times, while maintaining a relatively high degree of accuracy for the result.

It has been shown that summarizability is equivalent to the aggregation function being *distributive*, all paths being *strict*, and the hierarchies being *partitioning* in the relevant dimensions [8]. If data with time attached to it is aggregated such that data for one fact is only counted for one point in time, this result extends to hierarchies that are *snapshot strict* and *snapshot partitioning*. These concepts are formally defined below. In the definitions, we assume a dimension $D = (C, \leq)$.

Definition 2 If $\forall C_1, C_2 \in C(e_1, e_3 \in C_1 \wedge e_2 \in C_2 \wedge e_2 \leq e_1 \wedge e_2 \leq e_3 \Rightarrow e_1 = e_3)$ then the mapping between C_1 and C_2 is *strict*. Otherwise, it is *non-strict*. The hierarchy in dimension D is *strict* if all mappings in it are strict; otherwise, it is *non-strict*. Given a category $C_j \in D_i$, we say that there is a *strict path* from the set of facts F to C_j iff $\forall f \in F : f \rightsquigarrow e_1 \wedge f \rightsquigarrow e_2 \wedge e_1 \in C_j \wedge e_2 \in C_j \Rightarrow e_1 = e_2$[3]. The hierarchy in dimension D is *snapshot strict*, if at any given time t, the hierarchy is strict.

Definition 3 If $\forall C_1 \in C(C_1 \neq \top_D \wedge e_1 \in C_1 \Rightarrow \exists C_2 \in Pred(C_1)(\exists e_2 \in C_2(e_1 < e_2)))$, i.e., if every non-top value has a direct parent, we say that the hierarchy in dimension D is *partitioning*; otherwise, it is *non-partitioning*. The hierarchy in dimension D is *snapshot partitioning* if at any given time t, the hierarchy is partitioning.

Example 13 The hierarchy in the Residence dimension is strict and partitioning. The hierarchy in the Diagnosis dimension is non-strict and partitioning, but could have been non-partitioning. The sub-hierarchy of the Diagnosis dimension obtained by restriction to the standard classification is snapshot strict and snapshot partitioning.

4 The Algebra

This section defines an algebra on the multidimensional objects just defined. In line with the model definition, we first define the basic algebra and then define extensions for handling time and uncertainty. For some of the more complex operators, we provide examples of their use.

4.1 The Basic Algebra

We first define the fundamental operators. These are close to the standard relational algebra operators. For unary resp. binary operators, we assume a multidimensional object $M = (S, F, D = \{D_i\}, R = \{R_i\}), i = 1, .., n$ and multidimensional objects $M_k = (S_k, F_k, D_k = \{D_{ki_k}\}, R_k = \{R_{k_{i_k}}\}), k = 1, 2$. We note that the representations of the categories in the resulting MO's are the same as in the argument MO's, thus we do not specify the values representations for the resulting MO's. The aggregation types are only changed by the aggregate formation operator, so they are not specified for the other operators.

[3]Note that the paths from the set of facts F to the $\top_T$ categories are always strict.

For the operator definitions, we need some preliminary definitions. First, we define *Group*, that groups the facts characterized by the same dimension values together. Given an n-dimensional MO, $M = (\mathcal{S}, F, D = \{D_i\}, R = \{R_i\}), i = 1,..,n$, a set of categories $C = \{C_i \mid C_i \in D_i\}, i = 1,..,n$, one from each of the dimensions of M, and an n-tuple $(e_1,..,e_n)$, where $e_i \in C_i, i = 1,..,n$, we define *Group* as: $Group(e_1,..,e_n) = \{f \mid f \in F \wedge f \leadsto_1 e_1 \wedge .. \wedge f \leadsto_n e_n\}$.

Next, we define a *union* operator on dimensions, which performs union on the categories and the partial orders. Given two dimensions $D_1 = (C_1, \leq_1)$ and $D_2 = (C_2, \leq_2)$ of type $\mathcal{T}$, where $C_k = \{C_{kj}\}, k = 1,2,\ j = 1,..,m$, we define the union operator on dimensions, $\bigcup_D$, as: $D_1 \bigcup_D D_2 = (C', \leq')$, where $C' = \{C'_j\}, j = 1,..,m$, $C'_j = C_{1j} \cup C_{2j}$, where $\cup$ denotes regular set union, and $e_1 \leq' e_2 \Leftrightarrow e_1 \leq_1 e_2 \vee e_1 \leq_2 e_2$.

selection: Given a predicate p on the dimension types $\mathcal{D} = \{\mathcal{T}_i\}$, we define the selection σ as: $\sigma[p](M) = (\mathcal{S}', F', D', R')$, where $\mathcal{S}' = \mathcal{S}$, $F' = \{f \in F \mid \exists e_1 \in D_1,..,e_n \in D_n\ (p(e_1,..,e_n) \wedge f \leadsto_1 e_1 \wedge .. \wedge f \leadsto_n e_n)\}$, $D' = D$, $R' = \{R'_i\}$, and $R'_i = \{(f', e) \in R_i \mid f' \in F'\}$. Thus, we restrict the set of facts to those that are characterized by values where p evaluates to true. The fact-dimension relations are restricted accordingly, while the dimensions and the schema stay the same.

Example 14 If selection is applied to the "Patient" MO with the predicate $Name$ ="John Doe," the resulting MO has the same schema, the facts $F' = \{1\}$, the fact-dimension relations $R'_i = \{(1,e) \mid (1,e) \in R_i\}$, e.g., $R_2 = \{(1,9)\}$, and the dimension $D' = D$.

projection: Without loss of generality, we assume that the projection is over the k dimensions $D_1,..,D_k$. We then define the projection π as: $\pi[D_1,..,D_k](M) = (\mathcal{S}', F', D', R')$, where $\mathcal{S}' = (\mathcal{F}', \mathcal{D}')$, $\mathcal{F}' = \mathcal{F}$, $\mathcal{D}' = \{\mathcal{T}_1,..,\mathcal{T}_k\}$, $F' = F$, $D' = \{D_1,..,D_k\}$, and $R' = \{R_1,..,R_k\}$. Thus, we retain only the k dimensions, but the set of facts stays the same. Note that we do not remove "duplicate values." Thus the same combination of dimension values may be associated with several facts.

Example 15 If projection over the Name and Diagnosis dimensions is applied to the "Patient" MO, the resulting MO has the same fact type, only the Name and Diagnosis dimension types, the same set of facts, the Name and Diagnosis dimensions, and the fact-dimension relations for these two dimensions. A graphical illustration of the resulting MO is seen to the left in Figure 3.

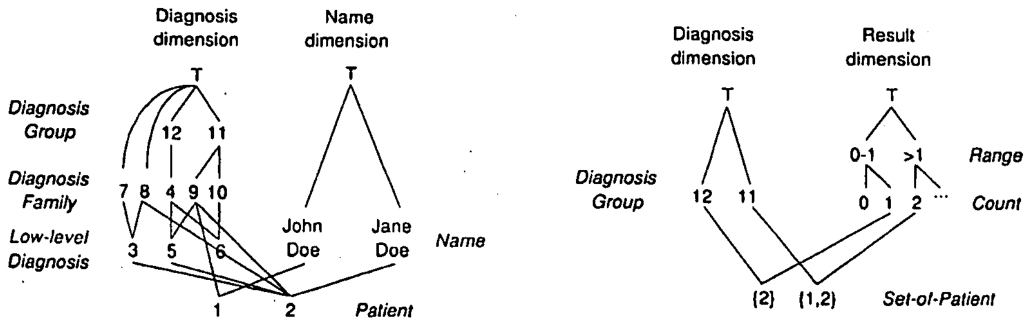


Figure 3: Resulting MO's for Projection and Aggregate Formation

rename: Given a multidimensional object, $M = (\mathcal{S}, F, D, R)$, and fact schema $\mathcal{S}' = (\mathcal{F}', \mathcal{D}')$, such that $\mathcal{D}$ is isomorphic with $\mathcal{D}'$, we define the rename ρ as: $\rho[\mathcal{S}'](M) = M'$, where $M' = (\mathcal{S}', F, D, R)$. We see that rename just return the contents of M with the new schema $\mathcal{S}'$, which has the same structure as the old schema $\mathcal{S}$. Rename is used to alter the names of dimensions so that dimensions with the same name, e.g., resulting from a "self-join," can be distinguished.

union: Given two n-dimensional MO's, $M_k = (\mathcal{S}_k, F_k, D_k, R_k), k = 1,2$ such that $\mathcal{S}_1 = \mathcal{S}_2$, we define the union $\bigcup$ as: $M_1 \bigcup M_2 = (\mathcal{S}', F', D', R')$, where $\mathcal{S}' = \mathcal{S}_1$, $F' = F_1 \cup F_2$, $D' = \{D_{1_i} \bigcup_D D_{2_i}, i = 1,..,n\}$, and $R' = \{R_{1_i} \cup R_{2_i}, i = 1,..,n\}$. In words, given two MO's with common schemas, we take the set union of the facts and the fact-dimension relations. The $\bigcup_D$ operator is used to combine the dimensions.

difference: Given two n-dimensional MO's ,$M_k = (\mathcal{S}_k, F_k, D_k, R_k), k = 1,2$ such that $\mathcal{S}_1 = \mathcal{S}_2$, we define the difference $\setminus$ as: $M_1 \setminus M_2 = (\mathcal{S}', F', D', R')$, where $\mathcal{S}' = \mathcal{S}_1$, $F' = F_1 \setminus F_2$, $D' = D_1$, $R' = \{R'_i, i = 1,..,n\}$, with $R'_i = \{(f', e) \mid f' \in F' \wedge (f', e) \in R_{1_i}$. Thus, given two MO's with common schemas, we take the set difference of the facts, the dimensions of the first argument MO are retained, and the fact-dimension relations are restricted to the new fact set. Note that we do not take the set difference of the dimensions, as this does not make sense.

Example 16 Performing the difference operator on the MO resulting from the projection example and the MO resulting from applying the selection $Name = ''\text{Jane Doe}''$ to the projection MO gives as a result an MO with the same schema, with the fact set $F = \{1\}$, the dimensions from the first argument, and the fact-dimension relations $R_1 = \{(1, 9)\}$ and $R_2 = \{(1, \text{John Doe})\}$.

identity-based join: Given two MO's, M_1 and M_2, and a predicate $p(f_1, f_2) \in \{f_1 = f_2, f_1 \neq f_2, true\}$, we define the identity-based join $\bowtie$ as: $M_1 \bowtie_{[p]} M_2 = (\mathcal{S}', F', D', R')$, where $(\mathcal{S}' = (\mathcal{F}', \mathcal{D}')$, $\mathcal{F}' = \mathcal{F}_1 \times \mathcal{F}_2$, $\mathcal{D}' = \mathcal{D}_1 \cup \mathcal{D}_2$, $F' = \{(f_1, f_2) \mid f_1 \in F_1 \wedge \ f_2 \in F_2 \wedge \ p(f_1, f_2)\}$, $D' = D_1 \cup D_2$, $R' = \{R'_i, i = 1,..,n_1 + n_2\}$, and $R'_i = \{(f', e) | f' = (f_1, f_2) \wedge f' \in F' \wedge ((i \leq n_1 \wedge (f_1, e) \in R_{1_i}) \vee (i > n_1 \wedge (f_2, e) \in R_{2_{i-n_1}}))\}$. In words, the new fact type is the type of *pairs* of the old fact types, and the new set of dimension types is the union of the old sets. The set of facts is the subset of the cross product of the old sets of facts where the join predicate p holds. For p equal to $f_1 = f_2$, $f_1 \neq f_2$, and *true*, the operation is an *equi-join*, *non-equi-join*, and *Cartesian product*, respectively. For the instance, the set of dimensions is the set union of the old sets of dimensions, and the fact-dimension relations relates a pair to an value if one member of the pair was related to that value before.

Example 17 We want to know if any patients are registered with more than one name. We take two copies of the "Patient" MO and perform projection over the Name dimension for both. For the second copy, the Name dimension type is renamed to "*Name2*". We then perform an identity-based join of the two with the predicate $f_1 = f_2$. This gives us an MO with two dimension types, *Name* and *Name2*. The fact type is the type of pairs of patients; the set of facts is still $F = \{1,2\}$, and the contents of the two dimensions are identical. The fact-dimension relations are also identical: $R_1 = \{(1, \text{John Doe}), (2, \text{Jane Doe})\}$ and $R_2 = \{(1, \text{John Doe}), (2, \text{Jane Doe})\}$. We can now perform a selection on this MO with the predicate $Name \neq Name2$ to find patients with more than one name.

aggregate formation: The aggregate formation operator is used to compute aggregate functions on the MO's. For notational convenience and following Klug [18], we assume the existence of a *family* of aggregation functions g that take some k-dimensional subset $\{D_{i_1}, .., D_{i_k}\}$ of the n dimensions as arguments, e.g., SUM_i sums the i-th dimension and SUM_{ij} sums the i-th and j-th dimensions. We assume a function $Args(g) = \{j \mid g \text{ uses dimension } j \text{ as argument}\}$ that returns the argument dimensions of g.

Given an n-dimensional MO, M, a dimension D_{n+1} of type $\mathcal{T}_{n+1}$, a function, $g : 2^F \mapsto D_{n+1}$[4] such that $g \in MIN\{Aggtype(\perp_{D_{i_j}}), j \in Args(g)\}$, and a set of categories $C_i \in D_i, i = 1,..,n$, we define aggregate formation, α, as: $\alpha[D_{n+1}, g, C_1,.., C_n](M) \doteq (\mathcal{S}', F', D', R')$, where $\mathcal{S}' = (\mathcal{F}', \mathcal{D}')$, $\mathcal{F}' = 2^{\mathcal{F}}$, $\mathcal{D}' = \{\mathcal{T}_i', i = 1,..,n\} \cup \{\mathcal{T}_{n+1}\}$, $\mathcal{T}_i' = (\mathcal{C}_i', \leq_{\mathcal{T}_i}', \perp_{\mathcal{T}_i}', \top_{\mathcal{T}_i}')$, $\mathcal{C}_i' = \{\mathcal{C}_{ij} \in \mathcal{T}_i \mid Type(C_i) \leq_{\mathcal{T}_i} \mathcal{C}_{ij}\}$, $\leq_{\mathcal{T}_i}' = \leq_{\mathcal{T}_i|_{\mathcal{C}_i'}}$, $\perp_{\mathcal{T}_i}' = Type(C_i)$, $\top_{\mathcal{T}_i}' = \top_{\mathcal{T}_i}$, $F' = \{Group(e_1,.., e_n) \mid (e_1,.., e_n) \in C_1 \times .. \times C_n \wedge Group(e_1,.., e_n) \neq \emptyset\}$, $D' = \{D_i', i = 1,..,n\} \cup \{D_{n+1}\}$, $D_i' = (C_i', \leq_i')$, $C_i' = \{C_{ij}' \in D_i \mid Type(C_{ij}') \in \mathcal{C}_i'\}$, $\leq_i' = \leq_{i|_{D_i'}}$, $R' = \{R_i', i = 1,..,n\} \cup \{R_{n+1}'\}$, $R_i' = \{(f', e_i') \mid \exists(e_1,.., e_n) \in C_1 \times .. \times C_n \, (f' = Group(e_1,.., e_n) \wedge f' \in F' \wedge e_i = e_i')\}$, and $R_{n+1}' = \cup_{(e_1,..,e_n) \in C_1 \times .. \times C_n} \{(Group(e_1,.., e_n), g(Group(e_1,.., e_n))) \mid Group(e_1,.., e_n) \neq \emptyset\}$. The aggregation types for the remaining parts of the argument dimensions are not changed. The aggregation types for the result dimension is given by the following rule. If g is distributive, the paths to $C_1,.., C_n$ are strict, and the hierarchies up to $C_1,.., C_n$ are partitioning, then $Aggtype(\perp_{D_{n+1}}) = MIN\{Aggtype(\perp_{D_j}), j \in Args(g)\}$. Otherwise, $Aggtype(\perp_{D_{n+1}}) = c$. For the higher categories in the result dimension, $Aggtype(C_m') = MIN\{Aggtype(C_m), Aggtype(\perp_{D_{n+1}})\}$.

Thus, for every combination $(e_1,.., e_n)$ of dimension values in the given "grouping" categories, we apply g to the set of facts $\{f\}$, where the f's are characterized by $(e_1,.., e_n)$, and place the result in the new dimension D_{n+1}. The facts are of type *sets* of the argument fact type, and the argument dimension types are restricted to the category types that are greater than or equal to the types of the given "grouping" categories. The dimension type for the result is added to the set of dimension types. The new set of facts consists of sets of facts, where the facts in a set share a combination of characterizing dimension values. The argument dimensions are restricted to the remaining category types, and the result dimension is added. The fact-dimension relations for the argument dimensions now link sets of facts directly to their corresponding combination of dimension values, and the fact-dimension relation for the result dimension links sets of facts to the function results for these sets. If the function g is distributive, the paths up to the grouping categories are strict, and the hierarchy up to the grouping categories is partitioning, i.e., g is "summarizable," then the aggregation type for the bottom category in the result dimension is the minimum of the aggregation types for the bottom categories in the dimensions that g uses as arguments ; otherwise, the aggregation type is c. For the higher categories, the minimum of the aggregation types given in the result dimension and the bottom category's aggregation type is used. Thus, aggregate results that are "unsafe" in the sense that they contain overlapping data, cannot be used for further aggregation. This prevents the user from getting incorrect results by accidentally "double-counting" data.

Example 18 We want to know the number of patients in each diagnosis group. To do so, we apply the aggregate-formation operator to the "Patient" MO with the *Diagnosis Group* category and the $\top$ categories from the other dimensions. The aggregate function g to be used is *set-count*, which counts the number of members in a set. The resulting MO has seven dimensions, but only the Diagnosis and Result dimensions are non-trivial, i.e., the remaining five dimensions contain only the $\top$ categories. The set of facts is still $F = \{1, 2\}$. The Diagnosis dimension is cut, so that only the part from *Diagnosis Group* and up is kept. The result dimension groups the counts into two ranges: "0–1" and ">1". The fact-dimension relation for the Diagnosis dimension links the sets of patients to their corresponding Diagnosis Group. The content is: $R_1 = \{(\{1, 2\}, 11), (\{2\}, 12)\}$, meaning that the sets of patients $\{1, 2\}$ and $\{2\}$ are characterized by diagnosis groups 11 and 12, respectively. The fact-dimension relation for the result dimension relate each group of patient to the count for the group. The content is: $R_7 = \{(\{1, 2\}, 2), (\{2, \}, 1)\}$, meaning that the results of g on the sets $\{1, 2\}$ and $\{2\}$ are 2 and 1, respectively. A graphical illustration of the MO, leaving out the trivial dimensions for simplicity, is seen on the right side of Figure 3. Note that each patient is only counted once for each diagnosis group, even though patient 2 has *several* diagnoses in each group.

[4]The function g "looks up" the required data for the facts in the relevant fact-dimension relations, e.g., SUM_i finds its data in fact-dimension relation R_i.

Now, we will show how other common OLAP and relational operators can be defined in terms of the fundamental operators.

value-based join: A join of two MO's on common dimension values can be made in the usual way by combining Cartesian product (a special case of the identity-based join), selection, and projection. *Natural join* is a special case of the value-based join, where the selection predicate requires that values from the "matching" dimensions should be equal, followed by projecting "out" the duplicate dimensions. Performing *drill-across* from one MO to another is just the value-based join of the two MO's on their common dimensions.

duplicate removal: We can remove "duplicate values," i.e., several facts characterized by the same combination of dimension values, by performing a *set-count* aggregate formation on the $\perp$ categories, followed by projecting out the result dimension.

SQL-like aggregation: Computation of an SQL aggregate function on an MO, grouped by a set of dimension categories, is done by first applying the aggregate formation operator to the MO with the given categories[5], and the given function. The dimensions not in the "GROUP BY" clause are then projected out.

star-join: A star-join as described in [4] is just a selection on the dimensions, usually combined with an aggregate formation with a given aggregate function on a set of category types.

drill-down: A drill-down on an MO means giving "more detail" by descending the dimension hierarchies. An implicit aggregation function, e.g., COUNT or SUM, is assumed. Thus, a drill-down corresponds to performing aggregate formation on "lower" category types with the given aggregate function. To get to the lower category types, a reference to the *original MO* is needed. In order to obtain the required detail, the aggregate formation is applied to the original object.

roll-up: A roll-up on an MO means giving "less detail" by ascending the dimension hierarchies, aggregating with an implicit aggregation function. This corresponds to performing aggregate formation on "higher" category types with the given aggregate function. Sometimes, we *also* need a reference to the original MO in this case. This is caused by the possible *non-summarizability* in the MO, which means that we cannot necessarily combine the aggregate results from intermediate levels into higher-level results, but need to compute the result directly from the lowest-level data (base data).

Theorem 1 *The algebra is closed.*

Proof: By examining the output of all operators, we see that the results are always MO's.

Theorem 2 *The algebra is at least as powerful as the relational algebra with aggregation functions* [18].

Proof: A relation r with schema $S_r = (a_1, .., a_n)$ is mapped to an n-dimensional MO $M = (\mathcal{S}, F, D, R)$, where $\mathcal{S} = (r, \{\mathcal{T}_i, i = 1, .., n\})$, $\mathcal{T}_i = (\{a_i, \top_{\mathcal{T}_i}\}, \leq_i, \top_{\mathcal{T}_i}, a_i)$, $a_i \leq_i a_i, a_i \leq_i \top_{\mathcal{T}_i}, \top_{\mathcal{T}_i} \leq_i \top_{\mathcal{T}_i}$, $F = \{(v_1, .., v_n) \in r\}$, $D = \{D_i\}, i = 1, .., n$, $D_i = (\{A_i, \top_i\}, \leq)$, $A_i = Dom(a_i)$, $\forall v \in A_i : v \leq \top$, $R = \{R_i\}, i = 1, .., n$, and $R_i = \{((v_1, .., v_i, .., v_n), v_i) \mid (v_1, .., v_i, .., v_n) \in r\}$. Thus, an n-ary relation is mapped to an MO with n "flat" dimensions, each containing the domain of the corresponding attribute.

[5]The categories not in the "GROUP BY" clause are the $\top$ categories of their dimensions.